Docket No.: 60188-721 **PATENT**

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of	:	Customer Number: 20277
Hidetoshi NARAHARA	:	Confirmation Number:
Serial No.:	:	Group Art Unit:
Filed: December 03, 2003	:	Examiner:

For: SIMULATION METHOD AND EMULATION METHOD

CLAIM OF PRIORITY AND
TRANSMITTAL OF CERTIFIED PRIORITY DOCUMENT

Mail Stop CPD
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 35 U.S.C. 119, Applicant hereby claim the priority of:

Japanese Patent Application No. JP2002-355819, filed on December 6, 2002.

cited in the Declaration of the present application. A certified copy is submitted herewith.

Respectfully submitted,

MCDERMOTT, WILL & EMERY

Michael E. Fogarty
Registration No. 36,139

600 13th Street, N.W.
Washington, DC 20005-3096
(202) 756-8000 MEF:gav
Facsimile: (202) 756-8087
Date: December 3, 2003

60188-1121
Hidetoshi NARAHARA
December 3, 2003

日 本 国 特 許 庁
JAPAN PATENT OFFICE

McDermott, Will & Emery

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: ２００２年１２月 ６日

出 願 番 号
Application Number: 特願２００２－３５５８１９

[ST. 10/C]: [ＪＰ２００２－３５５８１９]

出 願 人
Applicant(s): 松下電器産業株式会社

２００３年１０月２８日

特許庁長官
Commissioner,
Japan Patent Office 今井康夫

出証番号 出証特２００３－３０８８８７６

【書類名】　特許願
【整理番号】　5037540018
【提出日】　平成14年12月 6日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 17/50
【発明者】
　【住所又は居所】　大阪府門真市大字門真１００６番地　松下電器産業株式会社内
　【氏名】　楢原　英俊
【特許出願人】
　【識別番号】　000005821
　【氏名又は名称】　松下電器産業株式会社
【代理人】
　【識別番号】　100077931
　【弁理士】
　【氏名又は名称】　前田　弘
【選任した代理人】
　【識別番号】　100094134
　【弁理士】
　【氏名又は名称】　小山　廣毅
【選任した代理人】
　【識別番号】　100110939
　【弁理士】
　【氏名又は名称】　竹内　宏
【選任した代理人】
　【識別番号】　100110940
　【弁理士】
　【氏名又は名称】　嶋田　高久

【選任した代理人】

【識別番号】　100113262

【弁理士】

【氏名又は名称】　竹内　祐二

【選任した代理人】

【識別番号】　100115059

【弁理士】

【氏名又は名称】　今江　克実

【選任した代理人】

【識別番号】　100115510

【弁理士】

【氏名又は名称】　手島　勝

【選任した代理人】

【識別番号】　100115691

【弁理士】

【氏名又は名称】　藤田　篤史

【手数料の表示】

【予納台帳番号】　014409

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【包括委任状番号】　0006010

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　シミュレーション方法およびエミュレーション方法

【特許請求の範囲】

【請求項１】　記述ブロック単位に実行行履歴を記録する工程を有し、同一時刻に同じ記述ブロックが実行された場合、同一時刻の前回の実行行履歴を消去する工程を有するシミュレーション方法。

【請求項２】　記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のトレース結果から実行行を解析する工程を有するシミュレーション方法。

【請求項３】　記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号を単位時間単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号の単位時間単位のトレース結果から実行行を解析する工程を有するシミュレーション方法。

【請求項４】　記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をサイクル単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のサイクル単位のトレース結果から実行行を解析する工程を有するシミュレーション方法。

【請求項５】　記述ブロック単位にその記述ブロックの入力に対応するハードウェアエミュレーション処理での信号を抽出する工程を有し、記述ブロックの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各記述ブロックの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記記述ブロックの入力に対応するハードウェアエミュレ

ーション処理での信号の組み合わせと実行行の対応情報と前記記述ブロックの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有するエミュレーション方法。

【請求項６】　ロジックコーン単位にそのロジックコーンの入力に対応するハードウェアエミュレーション処理での信号を抽出する工程を有し、ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報と前記ロジックコーンの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有するエミュレーション方法。

【請求項７】　記述ブロック単位にその記述ブロック内の各行が実行される入力条件を解析する工程を有し、前記入力条件と実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記入力条件と実行行の対応情報と前記記述ブロックの入力信号のトレース結果から、実行行を解析する工程を有するシミュレーション方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

この発明はシミュレーション方法およびエミュレーション方法に関し、さらに詳しくは、半導体素子設計時のハードウェア記述言語の検証方法に関する。

【０００２】

【従来の技術】

近年、半導体素子の大規模化に伴い、半導体素子設計の効率化が重要視されている。このような背景の中、半導体素子の動作をプログラミング言語と同等に表現したハードウェア記述言語を用いて半導体素子を設計している。

【０００３】

ハードウェア記述を用いた半導体素子の設計では、設計したハードウェア記述が半導体素子の仕様通りに動作するか検証しなければならない。この検証を行う従来のシミュレーション装置の概略構成を図３２に示す。この装置は、シミュレーション実行処理部３と、実行行カウント処理部５とを備える。シミュレーション実行処理部３は、ＲＴＬ記述１をテストベクタ２に基づいてシミュレーションする。ＲＴＬ記述１は、半導体素子の機能をハードウェア記述言語で記述したものである。ＲＴＬとは、レジスタトランスファーレベルの略で、記述の抽象度の一種である。テストベクタ２は、設計者が半導体素子の仕様に合わせて作成したものである。設計者は、実行処理部３によるシミュレーション結果４が仕様と一致しているか逐一確認しながら、ＲＴＬ記述１が正しく設計されているか検証する。実行行カウント処理部５は、テストベクタ２の検証網羅性を測定するため、シミュレーション実行処理時のＲＴＬ記述１の実行行を逐一カウントし観測したカバレッジ結果６を出力する。設計者は、カバレッジ結果６からＲＴＬ記述１の未実行行を特定し、その未実行行が実行されるテストベクタ２を追加し再シミュレーションすることで検証漏れの無い品質の高い半導体素子を設計することができる。

【０００４】

【特許文献１】

特許第２６９９３７７号公報

【０００５】

【発明が解決しようとする課題】

図３２に示したシミュレーション装置では、シミュレーション実行処理部３が逐次処理であるため、機能上実行されていない行が瞬間的に実行されてしまい、実行行カウント処理部５において誤カウントが生じ、不当に網羅性の高いカバレッジ結果６が出力されるという問題があった。その様子を図３（ａ）のＲＴＬ記述例と図３（ｂ）のテストベクタ記述例を用いて説明する。

【０００６】

図３（ａ）および（ｂ）はハードウェア記述言語としてＶｅｒｉｌｏｇ－ＨＤＬを用いた例であり、左端の番号は記述の行番号を意味する。図３（ａ）におい

て、１００行および１１３行のａｌｗａｙｓ文は、定義されている信号が変化する度に実行される構文であり、１００行の場合、ａ，ｂ，ｃのどれかが変化する度に１０１～１１０行が実行される。このような、ある信号変化に伴って一連の処理を行う記述のかたまりを記述ブロックと定義する。図３（ａ）の場合、１００行から１１１行および１１３行から１１５行の２つの記述ブロックが存在する。

【０００７】

時刻０において図３（ｂ）の２０１行が実行され、ａとｂに０が代入される。次にａとｂの変化を受けて図３（ａ）の１００行と１１３行が評価されることになるのが逐次処理なので仮に１００行から実行されたとする。この時、ａ＝０，ｂ＝０であるがｃの値が確定されていないため、ｏｕｔへの代入は行われず、１１３行に処理が移りｃ＝１が代入される。次にｃの変化を受けて再び１００行が評価され、１０１行のｃａｓｅ文によって｛ａ，ｂ，ｃ｝＝｛０，０，１｝の１０３行が実行され、ｏｕｔに０が代入される。次に時刻１００において、図３（ｂ）の２０２行が実行され、ａ＝１が代入される。次にａの変化を受け、図３（ａ）の１００行が評価され、１０１行のｃａｓｅ文によって｛ａ，ｂ，ｃ｝＝｛１，０，１｝の１０７行が実行され、ｏｕｔに１が代入される。次に１１３行に処理が移りｃ＝０が代入される。次にこのｃの変化を受け、再び１００行が評価され、１０１行のｃａｓｅ文によって｛ａ，ｂ，ｃ｝＝｛１，０，０｝の１０６行が実行され、ｏｕｔに０が代入される。ここまでの処理におけるカバレッジ結果を図３３に示す。図３３は、図３（ａ）のＲＴＬ記述例の左端にカバレッジ結果として各行の実行回数を示したものである。上述のように逐次実行の度に実行行をカウントする場合には、機能的には実行されていない１０７行が実行されたことになり、不当に網羅性の高いカバレッジ結果になってしまう。

【０００８】

この発明の目的は、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐシミュレーション方法を提供することである。

【０００９】

【課題を解決するための手段】

この発明による第１のシミュレーション方法は、記述ブロック単位に実行行を記録し、同一時刻に同じ記述ブロックが実行された場合、同一時刻の前回の実行履歴を消去する工程を有する。

【００１０】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐことができる。

【００１１】

この発明による第２のシミュレーション方法は、記述ブロック単位にハードウエア記述を分解し、各記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のトレース結果から実行行を解析する工程を有する。

【００１２】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することができる。

【００１３】

この発明による第３のシミュレーション方法は、記述ブロック単位にハードウエア記述を分解し、各記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号を単位時間単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号の単位時間単位のトレース結果から実行行を解析する工程を有する。

【００１４】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐとともにオーバーヘッドを軽減することができる。

【００１５】

この発明による第４のシミュレーション方法は、記述ブロック単位にハードウエア記述を分解し、各記述ブロック単位にその記述ブロックの入力の組み合わせと実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をサイクル単位にてトレースする工程を有し、前記記述ブロックの入力の組み合わせと実行行の対応情報の解析結果と前記記述ブロックの入力信号のサイクル単位のトレース結果から実行行を解析する工程を有する。

【００１６】

上記シミュレーション方法によれば、設計する半導体素子の設計が完全同期設計である場合、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐシミュレーション方法とオーバーヘッドを軽減することができる。

【００１７】

この発明の第１のエミュレーション方法は、記述ブロック単位にハードウエア記述を分解し、各記述ブロック単位にその記述ブロックの入力に対応するハードウェアエミュレーション処理での信号を抽出する工程を有し、記述ブロックの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各記述ブロックの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記記述ブロックの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報と前記各記述ブロックの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有する。

【００１８】

上記エミュレーション方法によれば、ハードウェア記述言語のハードウェアエミュレーションにおけるカバレッジ測定を可能とすることができる。

【００１９】

この発明による第２のエミュレーション方法は、ロジックコーン単位にハードウエア記述を分解し、各ロジックコーン単位にそのロジックコーンの入力に対応

するハードウェアエミュレーション処理での信号を抽出する工程を有し、ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報を解析する工程を有し、エミュレーション実行処理工程にて、各ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号をトレースする工程を有し、前記ロジックコーンの入力に対応するハードウェアエミュレーション処理での信号の組み合わせと実行行の対応情報と前記ロジックコーンの入力信号に対応するハードウェアエミュレーション処理での信号をトレースした結果から実行行を解析する工程を有する。

【００２０】

上記エミュレーション方法によれば、ハードウェア記述言語のハードウェアエミュレーションにおいて、回路の最適化処理を実施した場合におけるカバレッジ測定を可能とする。

【００２１】

この発明による第５のシミュレーション方法は、記述ブロック単位にハードウエア記述を分解し、各記述ブロック単位にその記述ブロック内の各行が実行される入力条件を解析する工程を有し、前記入力条件と実行行の対応情報を解析する工程を有し、シミュレーション実行処理工程にて、各記述ブロックの入力信号をトレースする工程を有し、前記入力条件と実行行の対応情報と前記記述ブロックの入力信号のトレース結果から、実行行を解析する工程を有する。

【００２２】

上記シミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおいて、入力信号の多い記述ブロックのカバレッジ測定が可能となる。

【００２３】

【発明の実施の形態】

以下、この発明の実施の形態を図面を参照して詳しく説明する。なお、図中同一または相当部分には同一の符号を付しその説明は繰り返さない。

【００２４】

（第１の実施形態）

第１の実施形態によるシミュレーション装置の構成を図１に示す。このシミュ

レーション装置は、シミュレーション実行処理部３と、実行行カウント処理部１０とを備える。

【００２５】

シミュレーション実行処理部３は、ＲＴＬ記述１をテストベクタ２に基づいてシミュレーションする。ＲＴＬ記述１は、半導体素子の機能をハードウェア記述言語で記述したものである。ＲＴＬとは、レジスタトランスファーレベルの略で、記述の抽象度の一種である。テストベクタ２は、設計者が半導体素子の仕様に合わせて作成したものである。

【００２６】

実行行カウント処理部１０は、単位時間実行行トレース処理部１１と、実行行集計処理部１２とを含む。

【００２７】

単位時間実行行トレース処理部１１は、シミュレーション実行処理部３から実行行情報を受け取り、各記述ブロックについて単位時間ごとに実行行を抽出する。単位時間とは、設計者が指定できる最小時間を意味する。

【００２８】

実行行集計処理部１２は、単位時間実行行トレース処理部１１により抽出された実行行情報に基づいて各行の実行回数をカウントし、シミュレーション実行処理終了後にカバレッジ結果６を出力する。

【００２９】

次に、実行行カウント処理部１０における処理について図２を参照しつつ説明する。

【００３０】

シミュレーション実行処理部３によるシミュレーション実行処理の開始と同時に実行行カウント処理が開始される（ＳＴ１０１）。以後、シミュレーション実行処理が終了するまで実行行カウントループＳＴ１０２～ＳＴ１１１が繰り返され、シミュレーション時刻が更新されるまで単位時間実行行トレースループＳＴ１０３～ＳＴ１０８が繰り返される。

【００３１】

ステップＳＴ１０４において、シミュレーション実行処理部３から現在の実行行情報を受け取り、シミュレータの実行行が変化したか調べる。変化していない場合はシミュレーション時刻が更新されるまで再びステップＳＴ１０３のループに戻る。ステップＳＴ１０４において実行行が変化していた場合、現在の実行行が記述ブロックの先頭行か調べる（ＳＴ１０５）。記述ブロックの先頭行である場合、現在の実行行が含まれる記述ブロックのカウントテーブルをクリアする（ＳＴ１０６）。カウントテーブルとは、各記述ブロック単位に単位時間ごとの実行履歴を記憶しておくものである。次に現在の実行行をカウントテーブルに登録する（ＳＴ１０７）。ステップＳＴ１０５において記述ブロックの先頭行ではない場合は、カウントテーブルをクリアせずに、現在の実行行をカウントテーブルに登録する（ＳＴ１０７）。次にシミュレーション時刻が更新されるまで再びステップＳＴ１０３のループに戻る。ここまでのステップＳＴ１０３～ＳＴ１０８における処理は単位時間実行行トレース処理部１１において行われる。

【００３２】

シミュレーション時刻が更新された場合、カウントテーブルに登録されている全ての実行行の実行行集計テーブルを＋１する（ＳＴ１０９）。この処理は実行行集計処理１２において行われる。実行行集計テーブルとは、各行についてその行の実行回数を記録するものである。次にステップＳＴ１１０においてカウントテーブルをクリアする。シミュレーション実行処理が終了するまでステップＳＴ１０２の実行行カウント処理ループに戻り、実行行カウント処理を繰り返す。

【００３３】

図３（ａ）に示すＲＴＬ記述１と図３（ｂ）に示すテストベクタ２とに対して実行行カウント処理を行った場合のカバレッジ結果６を図４に示す。

【００３４】

以上のように第１の実施形態では、単位時間実行行トレース処理部１１により、単位時間単位に機能的に実行された行のみの実行行履歴を抽出するため、誤カウントの無い正確なカバレッジ結果６を得ることができる。

【００３５】

（第２の実施形態）

図３２に示した従来のシミュレーション装置ではシミュレーションの実行行をトレースする処理を実行行単位で行っているため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じ、シミュレーション速度が低下する。第２の実施形態では、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することを目的とする。

【００３６】

第２の実施形態によるシミュレーション装置の構成を図５に示す。このシミュレーション装置は、シミュレーション実行処理部３と、ブロック入力－実行行対応表作成処理部２０と、実行行カウント処理部２１とを備える。

【００３７】

ブロック入力－実行行対応表作成処理部２０は、ＲＴＬ記述１を記述ブロック単位に分解し、各記述ブロックに対してブロックの入力信号の組み合わせと実行行の対応を解析し、そのブロックの入力信号の組み合わせと実行行の対応情報を抽出する。

【００３８】

実行行カウント処理部２１は、ブロック入力トレース処理部２２と、実行行解析処理部２３と、実行行集計処理部１２とを含む。

【００３９】

ブロック入力トレース処理部２２は、ブロック入力－実行行対応表作成処理部２０により分解された記述ブロックの入力信号の変化毎に入力信号情報をシミュレーション実行処理部３から受け取る。

【００４０】

実行行解析処理部２３は、ブロック入力トレース処理部２２が受け取った入力信号情報と、ブロック入力－実行行対応表作成処理部２０が抽出したブロックの入力信号の組み合わせと実行行との対応情報とから実行行を抽出する。

【００４１】

実行行集計処理部１２は、実行行解析処理部２３により抽出された実行行情報を基に各行の実行回数をカウントし、シミュレーション実行処理終了後、カバレッジ結果６を出力する。

【００４２】

次に、ブロック入力－実行行対応表作成処理部２０で行われる処理について図６を参照しつつ説明する。

【００４３】

まず、ＲＴＬ記述１を読み込む（ＳＴ２０１）。ここでは図３（ａ）に示したＲＴＬ記述が読み込まれるものとする。

【００４４】

次に、読み込んだＲＴＬ記述１を記述ブロック単位に分割する（ＳＴ２０２）。この処理により、１００行から１１０行までの記述ブロックと１１３行から１１５行までの記述ブロックとに分割される（図３（ａ）参照）。続くステップＳＴ２０３～ＳＴ２０９のブロック別対応表作成ループにおける処理は、ステップＳＴ２０２で分解された全ての記述ブロックに対して行われる。

【００４５】

次に、処理対象の記述ブロックの入力信号を抽出する（ＳＴ２０４）。処理対象の記述ブロックが１１３～１１５行までの記述ブロックの場合、そのブロックの入力信号は、ａ，ｂ，ｃになる（図３（ａ）参照）。次からのステップＳＴ２０５～ＳＴ２０８のブロック入力パターン生成ループにおける処理は、ステップＳＴ２０４で抽出されたブロックの入力信号の組み合わせ全てに対して行われる。１１３～１１５行までの記述ブロックの場合、｛ａ，ｂ，ｃ｝＝｛０，０，０｝～｛１，１，１｝の８通りの組み合わせに対して処理されることになる。

【００４６】

次に、対象組み合わせパターンでの対象記述ブロックの実行行を解析する（ＳＴ２０６）。対象パターンが｛ａ，ｂ，ｃ｝＝｛０，０，０｝の場合、実行行は１００，１０１，１０２行となる（図３（ａ）参照）。

【００４７】

次に、ステップＳＴ２０６で得られたブロック入力と実行行の関係をブロック入力－実行行対応表に追加する（ＳＴ２０７）。

【００４８】

次に、ブロック入力の組み合わせを変えて、ステップＳＴ２０５～ＳＴ２０８

のループに戻る。全ての組み合わせの処理が終了後、次の記述ブロックに処理を移し、ステップＳＴ２０３～ＳＴ２０９のループに戻る。

【００４９】

以上の処理を図３（ａ）に示したＲＴＬ記述に対して行った場合のブロック入力－実行行対応表を図７に示す。

【００５０】

次に、実行行カウント処理部２１における実行行カウント処理について図８を参照しつつ説明する。

【００５１】

シミュレーション実行処理部３によるシミュレーション実行処理の開始と同時に実行行カウント処理を開始する（ＳＴ２５０）。

【００５２】

次に、ブロック入力－実行行対応表作成処理部２０で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する（ＳＴ２５１）。

【００５３】

次からのステップＳＴ２５２～ＳＴ２５６の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。

【００５４】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べる（ＳＴ２５３）。ここまでの処理はブロック入力トレース処理部２２において行われる。

【００５５】

ステップＳＴ２５３においてトレースポイントが変化していない場合はそのままステップＳＴ２５２のループへ戻る。トレースポイントが変化していた場合は、そのトレースポイントの値とブロック入力－実行行対応表作成処理部２０から得られたブロック入力－実行行対応表から実行行を抽出する（ＳＴ２５４）。この処理は実行行解析処理部２３において行われる。

【００５６】

次に、ステップＳＴ２５４で抽出された全ての実行行の実行行集計テーブルを＋１する（ＳＴ２５５）。この処理は実行行集計処理部１２において行われる。次にシミュレーション実行処理が終了するまで、ステップＳＴ２５２の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【００５７】

図３（ａ）に示したＲＴＬ記述１と図３（ｂ）に示したテストベクタ２とに対して上述の実行行カウント処理を行った場合のカバレッジ結果６は図３３に示した従来のカバレッジ結果と同等である。しかし、図３２に示した従来のシミュレーション装置ではシミュレーションの実行行をトレースする処理を実行行単位で行ってたのに対し、第２の実施形態ではブロック入力トレース処理２２により記述ブロックの入力信号のみをトレースするため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じシミュレーション速度が低下するという問題を解決できる。すなわち、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することができる。

【００５８】

（第３の実施形態）

図３２に示した従来のシミュレーション装置では、シミュレーション実行処理部３が逐次処理であるため、機能上実行されていない行が瞬間的に実行されてしまい、実行行カウント処理部５において誤カウントが生じ、不当に網羅性の高いカバレッジ結果６が出力されるおそれがある。また、シミュレーションの実行行をトレースする処理を実行行単位で行っているため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じ、シミュレーション速度が低下する。第３の実施形態では、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐとともにオーバーヘッドを軽減することを目的とする。

【００５９】

第３の実施形態によるシミュレーション装置の構成を図９に示す。このシミュレーション装置は、シミュレーション実行処理部３と、ブロック入力－実行行対応表作成処理部２０と、実行行カウント処理部３０とを備える。

【００６０】

実行行カウント処理部３０は、単位時間ブロック入力トレース処理部３１と、実行行解析処理部２３と、実行行集計処理部１２とを含む。

【００６１】

単位時間ブロック入力トレース処理部３１は、ブロック入力－実行行対応表作成処理部２０により分解された記述ブロックの入力信号の変化を単位時間毎に入力信号情報としてシミュレーション実行処理部３から受け取る。

【００６２】

実行行解析処理部２３は、単位時間ブロック入力トレース処理部３１が受け取った入力信号情報とブロック入力－実行行対応表作成処理部２０が抽出したブロックの入力信号の組み合わせと実行行の対応情報から実行行を抽出する。

【００６３】

次に、実行行カウント処理部３０における処理について図１０を参照しつつ説明する。

【００６４】

シミュレーション実行処理部３によるシミュレーション実行処理の開始と同時に実行行カウント処理を開始する（ＳＴ３００）。

【００６５】

次に、ブロック入力－実行行対応表作成処理部２０で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する（ＳＴ３０１）。

【００６６】

続くステップＳＴ３０２～ＳＴ３１０の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。また、ステップＳＴ３０３～ＳＴ３０６の単位時間トレース処理ループはシミュレーション時刻が更新されるまで繰り返される。

【００６７】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べる（ＳＴ３０４）。変化していない場合はそのままステップＳＴ３０３のループ

に戻る。トレースポイントが変化した場合、変化したトレースポイントが含まれる記述ブロックの単位時間変化テーブルを更新し、ステップＳＴ３０３のループに戻る（ＳＴ３０５）。単位時間変化テーブルとは、単位時間単位で各記述ブロックの入力信号の値を記憶するためのものである。

【００６８】

シミュレーション時刻が更新された場合、各記述ブロックの単位時間変化テーブルが更新されたか調べる（ＳＴ３０７）。ここまでの処理は単位時間ブロック入力トレース処理部３１において行われる。

【００６９】

ステップＳＴ３０７において単位時間変化テーブル更新されていない場合は、ステップＳＴ３０２のループに戻る。単位時間変化テーブルが更新さた場合は、その更新された単位時間変化テーブルの値とブロック入力－実行行対応表作成処理部２０から得られたブロック入力－実行行対応表から実行行を抽出する（ＳＴ３０８）。この処理は実行行解析処理部２３において行われる。

【００７０】

次に、ステップＳＴ３０８で抽出された全ての実行行の実行行集計テーブルを＋１する（ＳＴ３０９）。この処理は実行行集計処理部１２において行われる。以後、シミュレーション実行処理が終了するまでステップＳＴ３０２の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【００７１】

図３（ａ）に示したＲＴＬ記述と図３（ｂ）に示したテストベクタとに対して上述の実行行カウント処理を行った場合のカバレッジ結果６は図４に示したカバレッジ結果と同じである。

【００７２】

以上のように第３の実施形態では、単位時間ブロック入力トレース処理部３１により、単位時間単位に機能的に変化した記述ブロックの入力信号情報のみを抽出するため、誤カウントの無い正確なカバレッジ結果６を得ることができる。また、単位時間ブロック入力トレース処理部３１により記述ブロックの入力信号のみをトレースするため、実行行が更新される度にシミュレーション処理に対して

オーバーヘッドが生じシミュレーション速度が低下するという問題を解決できる。すなわち、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定のオーバーヘッドを軽減することができる。

【００７３】

（第４の実施形態）

図３２に示した従来のシミュレーション装置では、シミュレーション実行処理部３が逐次処理であるため、機能上実行されていない行が瞬間的に実行されてしまい、実行行カウント処理部５において誤カウントが生じ、不当に網羅性の高いカバレッジ結果６が出力されるおそれがある。また、シミュレーションの実行行をトレースする処理を実行行単位で行っているため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じ、シミュレーション速度が低下する。第４の実施形態では、設計する半導体素子の設計が完全同期設計である場合に、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐとともにオーバーヘッドを軽減することを目的とする。

【００７４】

第４の実施形態によるシミュレーション装置の構成を図１１に示す。このシミュレーション装置は、シミュレーション実行処理部３と、ブロック入力－実行行対応表作成処理部２０と、実行行カウント処理部４０とを備える。

【００７５】

実行行カウント処理部４０は、サイクル単位ブロック入力トレース処理部４１と、実行行解析処理部２３と、実行行集計処理部１２とを含む。

【００７６】

サイクル単位ブロック入力トレース処理部４１は、ブロック入力－実行行対応表作成処理部２０により分解された記述ブロックの入力信号の変化をサイクル時間毎に入力信号情報としてシミュレーション実行処理部３から受け取る。

【００７７】

実行行解析処理部２３は、サイクル単位ブロック入力トレース処理部４１が受け取った入力信号情報と、ブロック入力－実行行対応表作成処理部２０が抽出したブロックの入力信号の組み合わせと実行行の対応情報から実行行を抽出する。

【００７８】

次に、実行行カウント処理部４０における処理について図１２を参照しつつ説明する。

【００７９】

シミュレーション実行処理部３によるシミュレーション実行処理の開始と同時に実行行カウント処理を開始する（ＳＴ４００）。

【００８０】

次に、ブロック入力－実行行対応表作成処理部２０で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する（ＳＴ４０１）。

【００８１】

次からのステップＳＴ４０２～ＳＴ４１０の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。また、ステップＳＴ４０３～ＳＴ４０６の単位時間トレース処理ループはシミュレーション時刻が次のサイクルに更新されるまで繰り返される。

【００８２】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べる（ＳＴ４０４）。変化していない場合はそのままステップＳＴ４０３のループに戻る。トレースポイントが変化した場合は、変化したトレースポイントが含まれる記述ブロックのサイクル単位変化テーブルを更新し、ステップＳＴ４０３のループに戻る（ＳＴ４０５）。サイクル単位変化テーブルとは、サイクル単位で各記述ブロックの入力信号の値を記憶するためのものである。

【００８３】

シミュレーション時刻が次のサイクルに更新された場合、各記述ブロックのサイクル単位変化テーブルが更新されたか調べる（ＳＴ４０７）。ここまでの処理はサイクル単位ブロック入力トレース処理部４１において行われる。

【００８４】

ステップＳＴ４０７においてサイクル単位変化テーブル更新されていない場合は、ステップＳＴ４０２のループに戻る。サイクル単位変化テーブルが更新さた

場合は、その更新されたサイクル単位変化テーブルの値とブロック入力－実行行対応表作成処理部２０から得られたブロック入力－実行行対応表から実行行を抽出する（ＳＴ４０８）。この処理は実行行解析処理部２３において行われる。

【００８５】

次に、ステップＳＴ４０８で抽出された全ての実行行の実行行集計テーブルを＋１する（ＳＴ４０９）。この処理は実行行集計処理１２において行われる。以後、シミュレーション実行処理が終了するまでステップＳＴ４０２の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【００８６】

図３（ａ）に示したＲＴＬ記述と図３（ｂ）に示したテストベクタとに対して上述の実行行カウント処理を行った場合のカバレッジ結果６は図４に示したカバレッジ結果と同じである。

【００８７】

以上のように第４の実施形態では、サイクル単位ブロック入力トレース処理部４１により、サイクル単位に機能的に変化した記述ブロックの入力信号情報のみを抽出するため、誤カウントの無い正確なカバレッジ結果６を得ることができる。また、サイクル単位ブロック入力トレース処理部４１により記述ブロックの入力信号のみをトレースするため、実行行が更新される度にシミュレーション処理に対してオーバーヘッドが生じシミュレーション速度が低下するという問題を解決できる。また、実行行解析処理部２３と実行行集計処理部１２がサイクル単位でのみ実行となることから、第３の実施形態のシミュレーション方法と比べ、さらに処理速度を向上させることができる。

【００８８】

（第５の実施形態）

ハードウェア記述を用いた半導体素子の設計では、設計したハードウェア記述が半導体素子の仕様通りに動作するか検証するため、ソフトウェアベースのシミュレータの代わりにハードウェアエミュレータを用いる場合がある。ハードウェアエミュレータはソフトウェアベースのシミュレータと比べ、１万倍以上高速に検証することができる。次にハードウェアエミュレータの概要について説明する

。ハードウェアエミュレータでは、検証対象のＲＴＬ記述を論理構成処理によりゲートレベルのデータに変換し、それをＦＰＧＡ等の実際のハードウェアに置き換えることで高速にＲＴＬ記述の動作をエミュレートすることができる。図３（ａ）に示したＲＴＬ記述の場合、図１３に示すゲートレベルネットリストに置き換えられることになる。図１３のゲートレベルネットリストにおいて、Ｉ１～Ｉ５はインスタンス名であり、個々の回路素子に付けられた固有の名称である。Ｗ１～Ｗ７は回路素子間を繋ぐノードに対して付けられた固有の名称である。しかしながら、エミュレーション処理では行実行という概念が全く無いことから、カバレッジを測定することができなかった。

【００８９】

第５の実施形態では、ハードウェア記述言語のハードウェアエミュレーションにおけるカバレッジ測定を可能とすることを目的とする。

【００９０】

第５の実施形態によるエミュレーション装置の構成を図１４に示す。このエミュレーション装置は、論理合成部５１と、エミュレーション実行処理部５４と、ゲートレベルブロック入力－実行行対応表作成処理部５５と、実行行カウント処理部５６とを備える。

【００９１】

論理合成部５１は、ＲＴＬ記述１をハードウェアに置き換える為にゲートレベルネットリスト５２に変換する。図３（ａ）に示したＲＴＬ記述の場合、図１３のゲートレベルネットリストに変換される。また、ゲートレベルネットリスト変換時、ＲＴＬ記述１とゲートレベルネットリスト５２の対応を取るためのＲＴＬ－Ｇａｔｅ対応情報５３を出力する。図３（ａ）に示したＲＴＬ記述の場合のＲＴＬ－Ｇａｔｅ対応情報を図１５に示す。

【００９２】

エミュレーション実行処理部５４は、第１～第４の実施形態におけるシミュレーション実行処理部３に相当する機能を有する。

【００９３】

ゲートレベルブロック入力－実行行対応表作成処理部５５は、ＲＴＬ記述１を

記述ブロック単位に分解し、各記述ブロックに対してＲＴＬ－Ｇａｔｅ対応情報５３を用いて、エミュレーション実行処理部５４でのブロックの入力信号の組み合わせと実行行の対応を解析し、そのエミュレーション実行処理部５４でのブロックの入力信号の組み合わせと実行行の対応情報を抽出する。

【００９４】

実行行カウント処理部５６は、ゲートレベルブロック入力トレース処理部５７と、実行行解析処理部２３と、実行行集計処理部１２とを含む。

【００９５】

ゲートレベルブロック入力トレース処理部５７は、ゲートレベルブロック入力－実行行対応表作成処理部５５により分解された記述ブロックのエミュレーション実行処理部５４での入力信号の変化情報をエミュレーション実行処理部５４から受け取る。

【００９６】

実行行解析処理部２３は、ゲートレベルブロック入力トレース処理部５７が受け取った入力信号情報と、ゲートレベルブロック入力－実行行対応表作成処理部５５が抽出したエミュレーション実行処理部５４でのブロックの入力信号の組み合わせと実行行との対応情報とから実行行を抽出する。

【００９７】

実行行集計処理部１２は、実行行解析処理部２３より抽出した実行行情報を基に各行の実行回数をカウントし、エミュレーション実行処理終了後、カバレッジ結果６を出力する。

【００９８】

次に、ゲートレベルブロック入力－実行行対応表作成処理部５５における処理について図１６を参照しつつ説明する。

【００９９】

まずＲＴＬ記述１を読み込む（ＳＴ５０１）。ここでは図３（ａ）に示したＲＴＬ記述が読み込まれるものとする。

【０１００】

次に、ＲＴＬ－Ｇａｔｅ対応情報５３を読み込む（ＳＴ５０２）。

【０１０１】

次に、ステップＳＴ５０１で読み込んだＲＴＬ記述を記述ブロック単位に分割する（ＳＴ５０３）。この処理により、１００行から１１０行までの記述ブロックと１１３行から１１５行までの記述ブロックとに分割される（図３（ａ）参照）。続くステップＳＴ５０４～ＳＴ５１１のブロック別対応表作成ループにおける処理は、ステップＳＴ５０３で分解された全ての記述ブロックに対して行われる。

【０１０２】

次に、処理対象の記述ブロックの入力信号を抽出する（ＳＴ５０５）。処理対象の記述ブロックが１１３行から１１５行までの記述ブロックの場合、そのブロックの入力信号は、ａ，ｂ，ｃになる（図３（ａ）参照）。

【０１０３】

次に、ステップＳＴ５０５で抽出されたブロックの入力信号に対応するエミュレーション実行処理部５４での信号を抽出する（ＳＴ５０６）。これ以降、ブロックの入力信号に対応するエミュレーション実行処理部５４での信号のことを「ゲートレベルブロック入力」と呼ぶ。次からのステップＳＴ５０７～ＳＴ５１０のブロック入力パターン生成ループにおける処理は、ステップＳＴ５０５で抽出されたブロックの入力信号の組み合わせ全てに対して行われる。１１３行から１１５行までの記述ブロックの場合、｛ａ，ｂ，ｃ｝＝｛Ｗ１，Ｗ２，Ｗ４｝＝｛０，０，０｝～｛１，１，１｝の８通りの組み合わせに対して処理されることになる（図３（ａ）および図１５参照）。

【０１０４】

次に、対象組み合わせパターンでの対象記述ブロックの実行行を解析する（ＳＴ５０８）。対象パターンが｛ａ，ｂ，ｃ｝＝｛Ｗ１，Ｗ２，Ｗ４｝＝｛０，０，０｝の場合、実行行は１００，１０１，１０２行となる。

【０１０５】

次に、ステップＳＴ５０８で得られたブロック入力と実行行の関係をゲートレベルブロック入力－実行行対応表に追加する（ＳＴ５０９）。

【０１０６】

次に、ブロック入力の組み合わせを変えて、ステップＳＴ５０７～ＳＴ５１０のループに戻る。全ての組み合わせの処理が終了後、次の記述ブロックに処理を移し、ステップＳＴ５０４～ＳＴ５１１のループに戻る。

【０１０７】

以上の処理を図３（ａ）に示したＲＴＬ記述に対して行った場合のゲートレベルブロック入力－実行行対応表を図１７に示す。

【０１０８】

次に、実行行カウント処理部５６における処理について図１８を参照しつつ説明する。

【０１０９】

エミュレーション実行処理部５４によるエミュレーション実行処理の開始と同時に実行行カウント処理を開始する（ＳＴ５５０）。

【０１１０】

次に、ゲートレベルブロック入力－実行行対応表作成処理部５５で得られた各ブロックのゲートレベルブロック入力をエミュレーション実行処理におけるトレースポイントとして登録する（ＳＴ５５１）。次からのステップＳＴ５５２～ＳＴ５５６の実行行カウント処理ループはエミュレーション実行処理が終了するまで実行される。

【０１１１】

次に、エミュレーション実行処理においてトレースポイントが変化したか調べる（ＳＴ５５３）。ここまでの処理はゲートレベルブロック入力トレース処理部５７において行われる。

【０１１２】

ステップＳＴ５５３においてトレースポイントが変化していない場合はそのままステップＳＴ５５２のループに戻る。変化した場合は、その変化したトレースポイントの値とゲートレベルブロック入力－実行行対応表作成処理部５５から得られたゲートレベルブロック入力－実行行対応表から実行行を抽出する（ＳＴ５５４）。この処理は実行行解析処理部２３において行われる。

【０１１３】

次に、ステップＳＴ５５４で抽出された全ての実行行の実行行集計テーブルを＋１する（ＳＴ５５５）。この処理は実行行集計処理１２において行われる。

【０１１４】

これ以降、エミュレーション実行処理が終了するまでステップＳＴ５５２の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【０１１５】

図３（ａ）に示したＲＴＬ記述と図３（ｂ）に示したテストベクタに対して実行行カウント処理を行った場合のカバレッジ結果６は図４に示したカバレッジ結果と同じである。

【０１１６】

以上のように第５の実施形態では、ゲートレベルブロック入力－実行行対応表作成処理部５５から得られたゲートレベルブロック入力－実行行対応表とエミュレーション実行処理部５４のブロック入力に対応する信号をトレースする処理により、ＲＴＬ記述１の実行行を推定することができ、従来のエミュレーション方法では測定できなかったカバレッジ観測を実現することができる。また、エミュレーション実行処理はソフトウエアシミュレーション処理と異なり逐次処理では無いため、誤カウントの無い正確なカバレッジ結果６を得ることができる。

【０１１７】

（第６の実施形態）

図１４に示したエミュレーション装置において、論理合成処理部５１はＲＴＬ記述１をゲートレベルネットリスト５２に変換する。この際、ハードウェアに置き換える回路素子を少なくするため論理の最適化を行う場合がある。図３（ａ）に示したＲＴＬ記述に対して最適化処理を行った場合のゲートレベルネットリストを図１９に示す。この処理を行うことによりエミュレーション実行処理の対応回路規模を増加させることができ、エミュレーション処理速度も高速に行うことができる。しかしながら最適化処理を行うと、ＲＴＬ記述１の記述ブロックの入力に対応するゲートレベルブロック入力が最適化処理により削除され、第５の実施形態で説明したエミュレーション方法を適用できない問題が生じる。第６の実施形態では、ハードウェア記述言語のハードウェアエミュレーションにおいて回

路の最適化処理を実施した場合におけるカバレッジ測定を可能とすることを目的とする。

【０１１８】

第６の実施形態によるエミュレーション装置の構成を図２０に示す。このエミュレーション装置は、論理合成部５１と、エミュレーション実行処理部５４と、ロジックコーン入力－実行行対応表作成処理部６１と、実行行カウント処理部６２とを備える。

【０１１９】

論理合成処理部５１は、ＲＴＬ記述１をゲートレベルネットリスト５２に変換する際、ゲートレベルネットリスト５２の各ロジックコーンの入力に対応するＲＴＬ記述の信号情報（ロジックコーンＲＴＬ－Ｇａｔｅ対応情報）６０を出力する。ロジックコーンとは、論理回路をフリップフロップなどの記憶素子と組み合わせ回路に分割した際、各記憶素子の入力または回路の出力端子を起点として、その起点に影響を与える組み合わせ回路を意味する。図３（ａ）に示したＲＴＬ記述に対して論理合成処理を行った際のロジックコーンＲＴＬ－Ｇａｔｅ対応情報６０を図２１に示す。

【０１２０】

ロジックコーン入力－実行行対応表作成処理部６１は、ロジックコーンＲＴＬ－Ｇａｔｅ対応情報６０を用いて、ＲＴＬ記述１をロジックコーン単位に分解し、各ロジックコーン単位にエミュレーション実行処理部５４でのロジックコーン入力信号の組み合わせとＲＴＬ記述１の実行行の対応情報を抽出する。

【０１２１】

実行行カウント処理部６２は、ロジックコーン入力トレース処理部６３と、実行行解析処理部２３と、実行行集計処理部１２とを含む。

【０１２２】

ロジックコーン入力トレース処理部６３は、エミュレーション実行処理部５４でのロジックコーン入力の変化情報をエミュレーション実行処理部５４から受け取る。

【０１２３】

実行行解析処理部２３は、ロジックコーン入力トレース処理部６３が受け取った入力信号情報と、ロジックコーン入力－実行行対応表作成処理部６１が抽出したエミュレーション実行処理部５４でのロジックコーンの入力信号の組み合わせと実行行との対応情報とから実行行を抽出する。

【０１２４】

実行行集計処理部１２は、実行行解析処理部２３より抽出した実行行情報を基に各行の実行回数をカウントし、エミュレーション実行処理終了後、カバレッジ結果６を出力する。

【０１２５】

次に、ロジックコーン入力－実行行対応表作成処理部６１における処理について図２２を参照しつつ説明する。

【０１２６】

まずＲＴＬ記述１を読み込む（ＳＴ６０１）。ここでは図３（ａ）に示したＲＴＬ記述を読み込むものとする。

【０１２７】

次に、ロジックコーンＲＴＬ－Ｇａｔｅ対応情報６０を読み込む（ＳＴ６０２）。

【０１２８】

次に、ステップＳＴ６０１で読み込んだＲＴＬ記述１をロジックコーン単位に分割する（ＳＴ６０３）。図３（ａ）のＲＴＬ記述１の場合、１００行から１１５行全てが１つのロジックコーンに相当する。

【０１２９】

次からのステップＳＴ６０４～ＳＴ６１１のロジックコーン別対応表作成ループにおける処理は、ステップＳＴ６０３で分解された全てのロジックコーンに対して行われる。

【０１３０】

次に、処理対象のロジックコーンの入力信号を抽出する（ＳＴ６０５）。図３（ａ）のＲＴＬ記述１の場合、ロジックコーンの入力信号は、ａ，ｂになる。

【０１３１】

次に、ステップＳＴ６０５で抽出されたロジックコーンの入力信号に対応するエミュレーション実行処理部５４での信号を抽出する（ＳＴ６０６）。これ以降、ロジックコーン入力信号に対応するエミュレーション実行処理部５４での信号のことを「ゲートレベルロジックコーン入力」と呼ぶ。

【０１３２】

次からのステップＳＴ６０７～ＳＴ６１０のロジックコーン入力パターン生成ループにおける処理は、ステップＳＴ６０５で抽出されたロジックコーンの入力信号の組み合わせ全てに対して行われる。図３（ａ）のＲＴＬ記述１の場合、｛ａ，ｂ｝＝｛Ｗ１，Ｗ２｝＝｛０，０｝～｛１，１｝の４通りの組み合わせに対して処理されることになる。

【０１３３】

次に、対象組み合わせパターンでの対象ロジックコーンの実行行を解析する（ＳＴ６０８）。対象パターンが｛ａ，ｂ｝＝｛Ｗ１，Ｗ２｝＝｛０，０｝の場合、実行行は１００，１０１，１０２，１１３，１１４行となる。

【０１３４】

次に、ステップＳＴ６０８で得られたゲートレベルロジックコーン入力と実行行の関係をロジックコーン入力－実行行対応表に追加する（ＳＴ６０９）。

【０１３５】

次に、ロジックコーンの入力の組み合わせを変えて、ステップＳＴ６０７～ＳＴ６１０のループに戻る。全ての組み合わせの処理が終了後、次のロジックコーンに処理を移し、ステップＳＴ６０４～ＳＴ６１１のループに戻る。以上の処理を図３（ａ）のＲＴＬ記述１に対して行った場合のロジックコーン入力－実行行対応表を図２３に示す。

【０１３６】

次に、実行行カウント処理部６２における処理について図２４を参照しつつ説明する。

【０１３７】

エミュレーション実行処理開始と同時に実行行カウント処理を開始する（ＳＴ６５０）。

【０１３８】

次に、ロジックコーン入力－実行行対応表作成処理部６１で得られた各ロジックコーンのゲートレベルロジックコーン入力をエミュレーション実行処理におけるトレースポイントとして登録する（ＳＴ６５１）。

【０１３９】

次からのステップＳＴ６５２～ＳＴ６５６の実行行カウント処理ループはエミュレーション実行処理が終了するまで実行される。

【０１４０】

次に、エミュレーション実行処理においてトレースポイントが変化したか調べる（ＳＴ６５３）。ここまでの処理はロジックコーン入力トレース処理６３において行われる。

【０１４１】

ステップＳＴ６５３においてトレースポイントが変化していない場合はそのままステップＳＴ６５２のループに戻る。変化した場合は、その変化したトレースポイントの値とロジックコーン入力－実行行対応表作成処理部６１から得られたロジックコーン入力－実行行対応表から実行行を抽出する（ＳＴ６５４）。この処理は実行行解析処理部２３において行われる。

【０１４２】

次に、ステップＳＴ６５４で抽出された全ての実行行の実行行集計テーブルを＋１する（ＳＴ６５５）。この処理は実行行集計処理部１２において行われる。

【０１４３】

これ以降、エミュレーション実行処理が終了するまでステップＳＴ６５２の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【０１４４】

図３（ａ）のＲＴＬ記述１と図３（ｂ）のテストベクタとに対して実行行カウント処理を行った場合のカバレッジ結果６は図４に示したカバレッジ結果と同じである。

【０１４５】

以上のように第６の実施形態では、ロジックコーンの入力は、論理合成処理部

５１による最適化処理を実行した場合でも必ずゲートレベルネットリスト５２に存在するため、論理合成処理部５１による最適化処理を実行した場合でもカバレッジを測定することができる。したがって、カバレッジを測定する場合でも最適化処理を実行でき、第５の実施形態のエミュレーション方法と比べ、大規模回路に対応することができる。また、エミュレーション処理速度も向上させることができる。

【０１４６】

（第７の実施形態）

第２～第４の実施形態によるシミュレーション装置では、ブロック入力－実行行対応表作成処理部２０は記述ブロック単位の各入力信号をｂｉｔ単位で扱うため、記述ブロックの入力信号の数が多くなるとその入力信号全ての組み合わせが膨大な数になり処理できなくなるおそれがある。第７の実施形態では、ハードウェア記述言語のシミュレーションにおいて、入力信号の多い記述ブロックのカバレッジ測定を可能にすることを目的とする。

【０１４７】

第７の実施形態によるシミュレーション装置の構成を図２５に示す。このシミュレーション装置は、シミュレーション実行処理部３と、条件別ブロック入力－実行行対応表作成処理部７０と、実行行カウント処理部７１とを備える。

【０１４８】

条件別ブロック入力－実行行対応表作成処理部７０は、ＲＴＬ記述１を記述ブロック単位に分解し、各記述ブロックに対してブロックの入力信号の条件と実行行の対応を解析し、そのブロックの入力信号の条件と実行行の対応情報を抽出する。第２～第４の実施形態におけるブロック入力－実行行対応表作成処理部２０ではそのブロックの入力をビット単位で扱っていたのに対し、条件別ブロック入力－実行行対応表作成処理部７０では各行の実行条件を解析し、ブロックの入力をその条件単位で扱う点が異なる。

【０１４９】

実行行カウント処理部７１は、ブロック入力トレース処理部２２と、条件別実行行解析処理部７２と、実行行集計処理部１２とを含む。

【０１５０】

ブロック入力トレース処理部２２は、条件別ブロック入力－実行行対応表作成処理部７０により分解された記述ブロックの入力信号の変化毎に入力信号情報をシミュレーション実行処理部３から受け取る。

【０１５１】

条件別実行行解析処理部７２は、ブロック入力トレース処理部２２が受け取った入力信号情報と、条件別ブロック入力－実行行対応表作成処理部７０が抽出したブロックの入力条件と実行行の対応情報とから実行行を抽出する。

【０１５２】

実行行集計処理部１２は、実行行解析処理部２３により抽出された実行行情報を基に各行の実行回数をカウントし、シミュレーション実行処理終了後、カバレッジ結果６を出力する。

【０１５３】

次に、条件別ブロック入力－実行行対応表作成処理部７０における処理について図２６を参照しつつ説明する。

【０１５４】

まずＲＴＬ記述１を読みこむ（ＳＴ７０１）。ここでは図２７に示すＲＴＬ記述１を読み込むものとする。

【０１５５】

次に、読みこんだＲＴＬ記述１を記述ブロック単位に分割する（ＳＴ７０２）。この処理により、３００行から３０４行までの記述ブロックと３０６行から３０８行までの記述ブロックとに分割される（図２７参照）。

【０１５６】

次からのステップＳＴ７０３～ＳＴ７１０のブロック別対応表作成ループにおける処理は、ステップＳＴ７０２で分解された全ての記述ブロックに対して行われる。また、ステップＳＴ７０４～ＳＴ７０９の実行条件解析ループは、処理対象の記述ブロックの各行に対して解析が終了するまで繰り返される。

【０１５７】

次に、処理対象行の実行条件を解析する（ＳＴ７０５）。

【０１５８】

次に、ステップＳＴ７０５で抽出した実行条件が、条件別ブロック入力－実行行対応表に存在するか調べる（ＳＴ７０６）。存在する場合は、条件別ブロック入力－実行行対応表の同条件に現在の処理対象行を実行行として追加する（ＳＴ７０７）。ステップＳＴ７０６において条件別ブロック入力－実行行対応表に存在しない場合は、条件別ブロック入力－実行行対応表に実行条件と実行行を追加する（ＳＴ７０８）。

【０１５９】

すべての記述ブロック内の行に対して処理を終了した後（ＳＴ７０９）、次の記述ブロックに対象ブロックを移し（ＳＴ７１０）、ブロック別対応表作成ループＳＴ７０３に戻る。

【０１６０】

以上の処理を図２７のＲＴＬ記述１に対して行った場合の条件別ブロック入力－実行行対応表を図２８に示す。

【０１６１】

次に、実行行カウント処理部７１における処理について図２９を参照しつつ説明する。

【０１６２】

シミュレーション実行処理開始と同時に実行行カウント処理を開始する（ＳＴ７５０）。

【０１６３】

次に、条件別ブロック入力－実行行対応表作成処理部７０で得られた各ブロックの入力信号をシミュレーション実行処理におけるトレースポイントとして登録する（ＳＴ７５１）。

【０１６４】

次からのステップＳＴ７５２～ＳＴ７５６の実行行カウント処理ループはシミュレーション実行処理が終了するまで実行される。

【０１６５】

次に、シミュレーション実行処理においてトレースポイントが変化したか調べ

る（ＳＴ７５３）。ここまでの処理はブロック入力トレース処理部２２において行われる。

【０１６６】

ステップＳＴ７５３においてトレースポイントが変化していない場合はそのままステップＳＴ７５２のループへ戻る。トレースポイントが変化していた場合は、そのトレースポイントの値と条件別ブロック入力－実行行対応表作成処理部７０から得られた条件別ブロック入力－実行行対応表から該当条件の実行行を抽出する（ＳＴ７５４）。この処理は条件別実行行解析処理部７２において行われる。

【０１６７】

次に、ステップＳＴ７５４で抽出された全ての実行行の実行行集計テーブルを＋１する（ＳＴ７５５）。この処理は実行行集計処理１２において行われる。

【０１６８】

これ以降、シミュレーション実行処理が終了するまで、ステップＳＴ７５２の実行行カウント処理ループに戻って、実行行カウント処理を繰り返す。

【０１６９】

図２８に示したＲＴＬ記述１と図３０に示すテストベクタ２とに対して実行行カウント処理を行った場合のカバレッジ結果６を図３１に示す。

【０１７０】

第７の実施形態ではカバレッジの誤カウントを防止する処理を実施していない。したがってカバレッジ結果（図３１）は誤カウントの含まれる結果となっている。しかしながら、第３～第４の実施形態において説明したカバレッジの誤カウントを防止する処理を適用すれば誤カウントの無い正確なカバレッジ結果を得ることができる。

【０１７１】

第２～第４の実施形態では記述ブロック入力と実行行の対応をｂｉｔ単位で扱っているのに対し、第７の実施形態では条件別ブロック入力実行行対応表作成処理部７０が記述ブロックの各行の実行条件単位で入力条件と実行行の対応をとるため、入力信号の多い記述ブロックに対してカバレッジ測定が可能となる。

【０１７２】

【発明の効果】

この発明によるシミュレーション方法によれば、ハードウェア記述言語のシミュレーションにおけるカバレッジ測定の誤カウントを防ぐことができる。

【図面の簡単な説明】

【図１】　第１の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図２】　図１に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図３】　（ａ）はＲＴＬ記述の一例を示す図であり、（ｂ）はテストベクタの一例を示す図である。

【図４】　カバレッジ結果を示す図である。

【図５】　第２の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図６】　図５に示したブロック入力－実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図７】　ブロック入力－実行行対応表の一例を示す図である。

【図８】　図５に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図９】　第３の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図１０】　図９に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図１１】　第４の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図１２】　図１１に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図１３】　ゲートレベルネットリストの一例を示す図である。

【図１４】　第５の実施形態によるエミュレーション装置の構成を示すブロ

ック図である。

【図１５】　ＲＴＬ－Ｇａｔｅ対応情報の一例を示す図である。

【図１６】　図１４に示したゲートレベルブロック入力－実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図１７】　ゲートレベルブロック入力－実行行対応表の一例を示す図である。

【図１８】　図１４に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図１９】　ゲートレベルネットリストの一例を示す図である。

【図２０】　第６の実施形態によるエミュレーション装置の構成を示すブロック図である。

【図２１】　ロジックコーンＲＴＬ－Ｇａｔｅ対応情報の一例を示す図である。

【図２２】　図２０に示したロジックコーン入力－実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図２３】　ロジックコーン入力－実行行対応表の一例を示す図である。

【図２４】　図２０に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図２５】　第７の実施形態によるシミュレーション装置の構成を示すブロック図である。

【図２６】　図２５に示した条件別ブロック入力－実行行対応表作成処理部における処理の流れを示すフローチャートである。

【図２７】　ＲＴＬ記述の一例を示す図である。

【図２８】　条件別ブロック入力－実行行対応表の一例を示す図である。

【図２９】　図２５に示した実行行カウント処理部における処理の流れを示すフローチャートである。

【図３０】　テストベクタの一例を示す図である。

【図３１】　カバレッジ結果を示す図である。

【図３２】　従来のシミュレーション装置の構成を示すブロック図である。

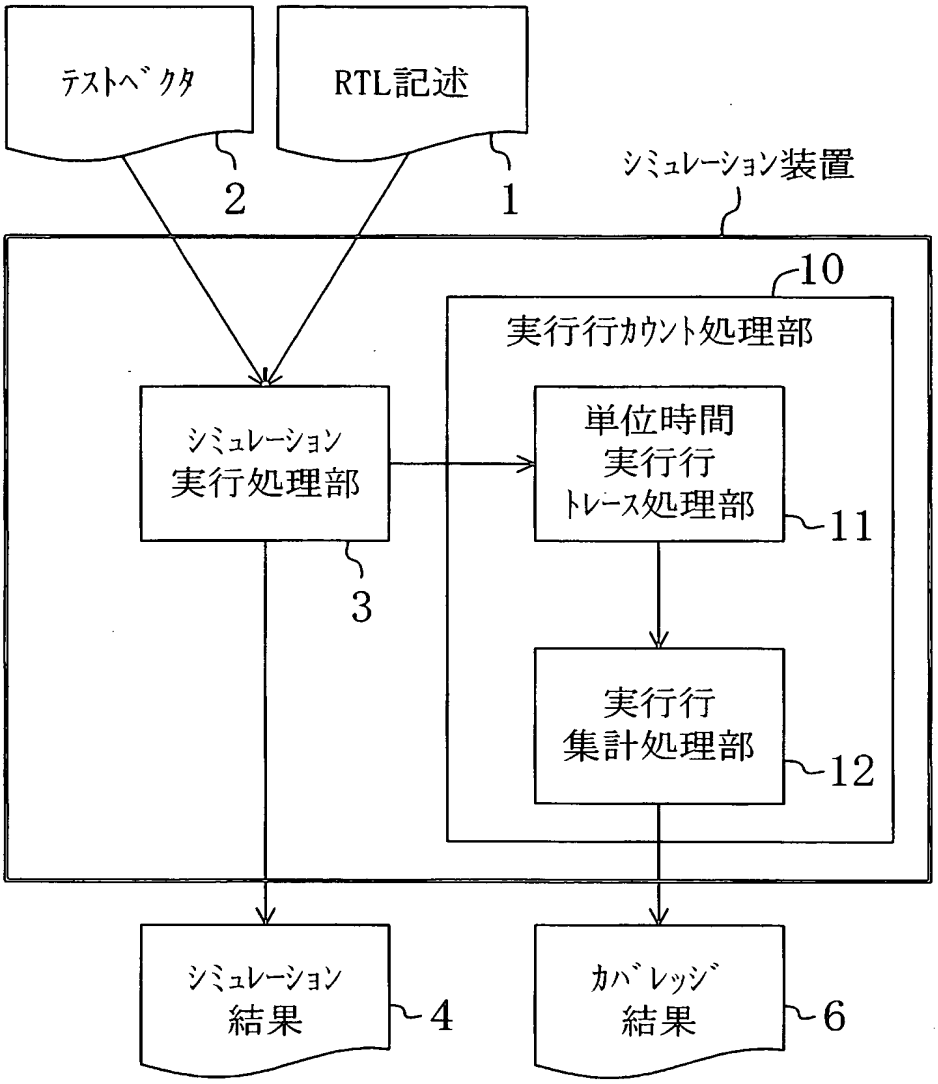
【図３３】　カバレッジ結果を示す図である。

【符号の説明】

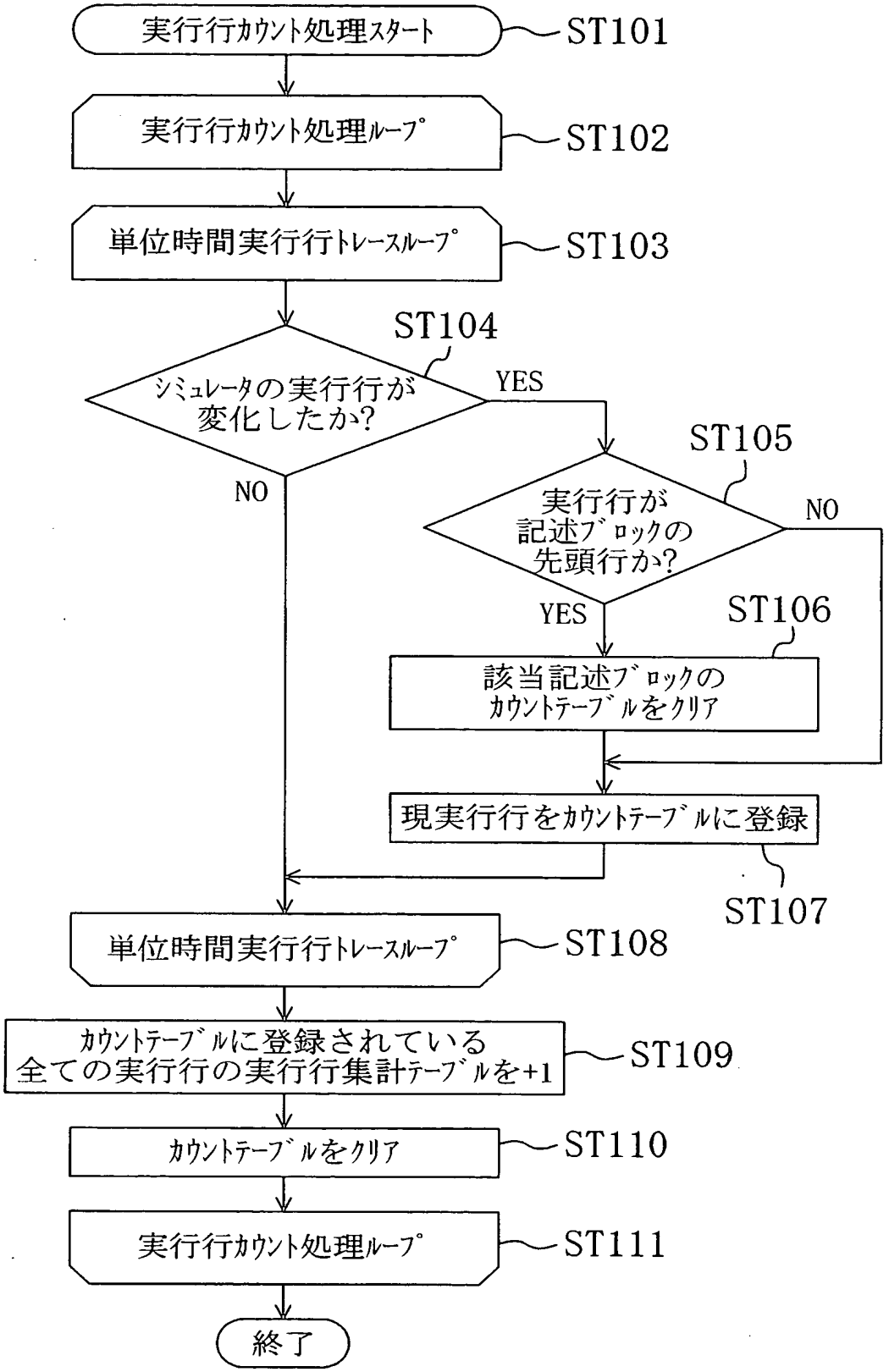
１　ＲＴＬ記述、２　テストベクタ、３　シミュレーション実行処理部、４　シミュレーション結果、５，１０，２１，３０，４０，５６，６２，７１　実行行カウント処理部、６　カバレッジ結果、１１　単位時間実行行トレース処理部、１２　実行行集計処理部、２０　ブロック入力－実行行対応表作成処理部、２２　ブロック入力トレース処理部、２３　実行行解析処理部、３１　単位時間ブロック入力トレース処理部、４１　サイクル単位ブロック入力トレース処理部、５１　論理合成処理部、５２　ゲートレベルネットリスト、５３　ＲＴＬ－Ｇａｔｅ対応情報、５４　エミュレーション実行処理部、５５　ゲートレベルブロック入力－実行行対応表作成処理部、５７　ゲートレベルブロック入力トレース処理部、６０　ロジックコーンＲＴＬ－Ｇａｔｅ対応情報、６１　ロジックコーン入力－実行行対応表作成処理部、６３　ロジックコーン入力トレース処理部、７０　条件別ブロック入力－実行行対応表作成処理部、７２　条件別実行行解析処理部。

【書類名】　図面

【図１】


テストベクタ
RTL記述
2
1
シミュレーション装置
10
実行行カウント処理部
シミュレーション
実行処理部
単位時間
実行行
トレース処理部
11
3
実行行
集計処理部
12
シミュレーション
結果
4
カバレッジ
結果
6

【図２】


実行行カウント処理スタート
ST101
実行行カウント処理ループ
ST102
単位時間実行行トレースループ
ST103
ST104
シミュレータの実行行が変化したか?
YES
NO
ST105
実行行が記述ブロックの先頭行か?
NO
YES
ST106
該当記述ブロックのカウントテーブルをクリア
現実行行をカウントテーブルに登録
ST107
単位時間実行行トレースループ
ST108
カウントテーブルに登録されている全ての実行行の実行行集計テーブルを+1
ST109
カウントテーブルをクリア
ST110
実行行カウント処理ループ
ST111
終了

【図３】

1

(a)

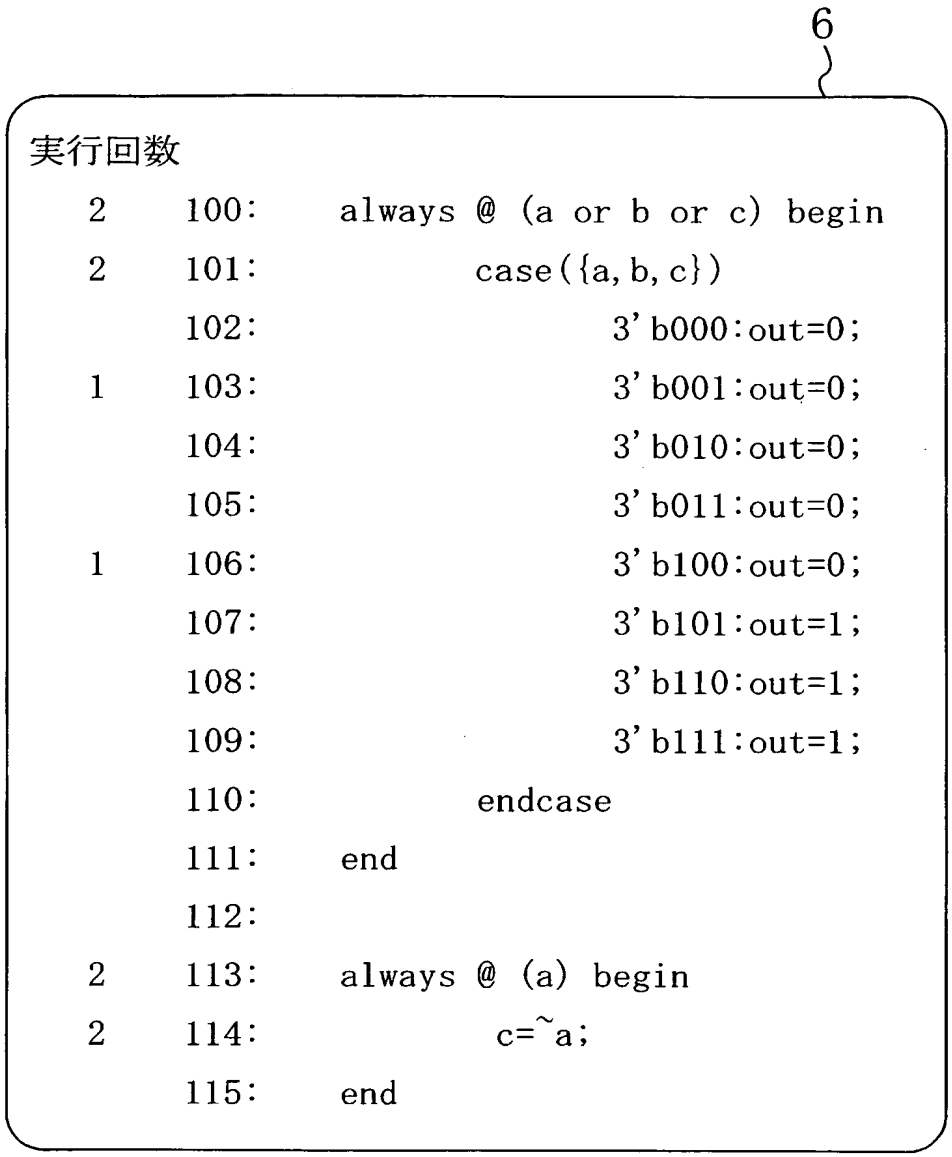
```
100:    always @ (a or b or c) begin
101:            case({a,b,c})
102:                    3'b000:out=0;
103:                    3'b001:out=0;
104:                    3'b010:out=0;
105:                    3'b011:out=0;
106:                    3'b100:out=0;
107:                    3'b101:out=1;
108:                    3'b110:out=1;
109:                    3'b111:out=1;
110:             endcase
111:    end
112:
113:    always @ (a) begin
114:            c=~a;
115:    end
```

2

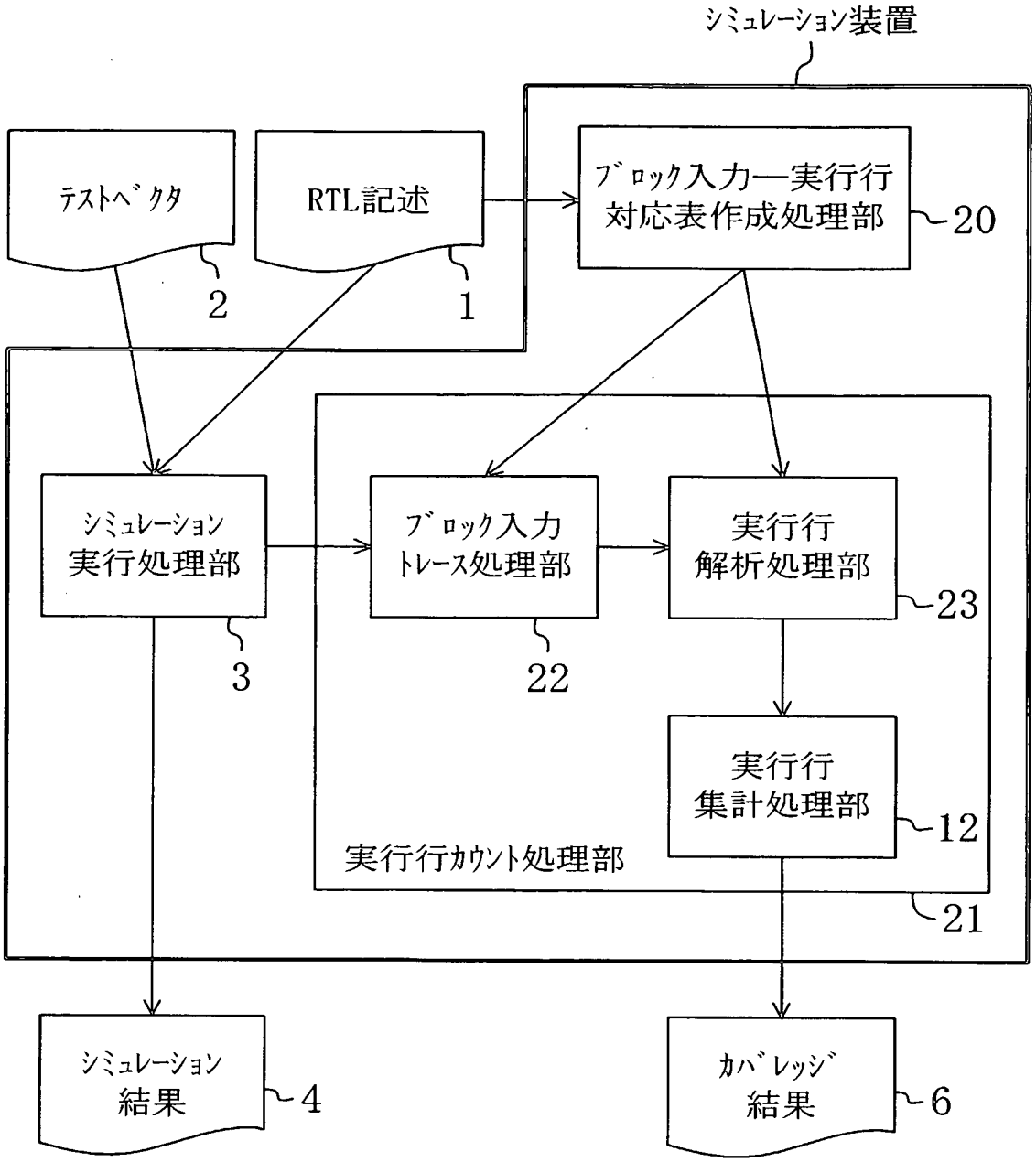
(b)

```
200:    initial begin
201:            #0      a=0; b=0;
202:            #100    a=1;
203:     end
```

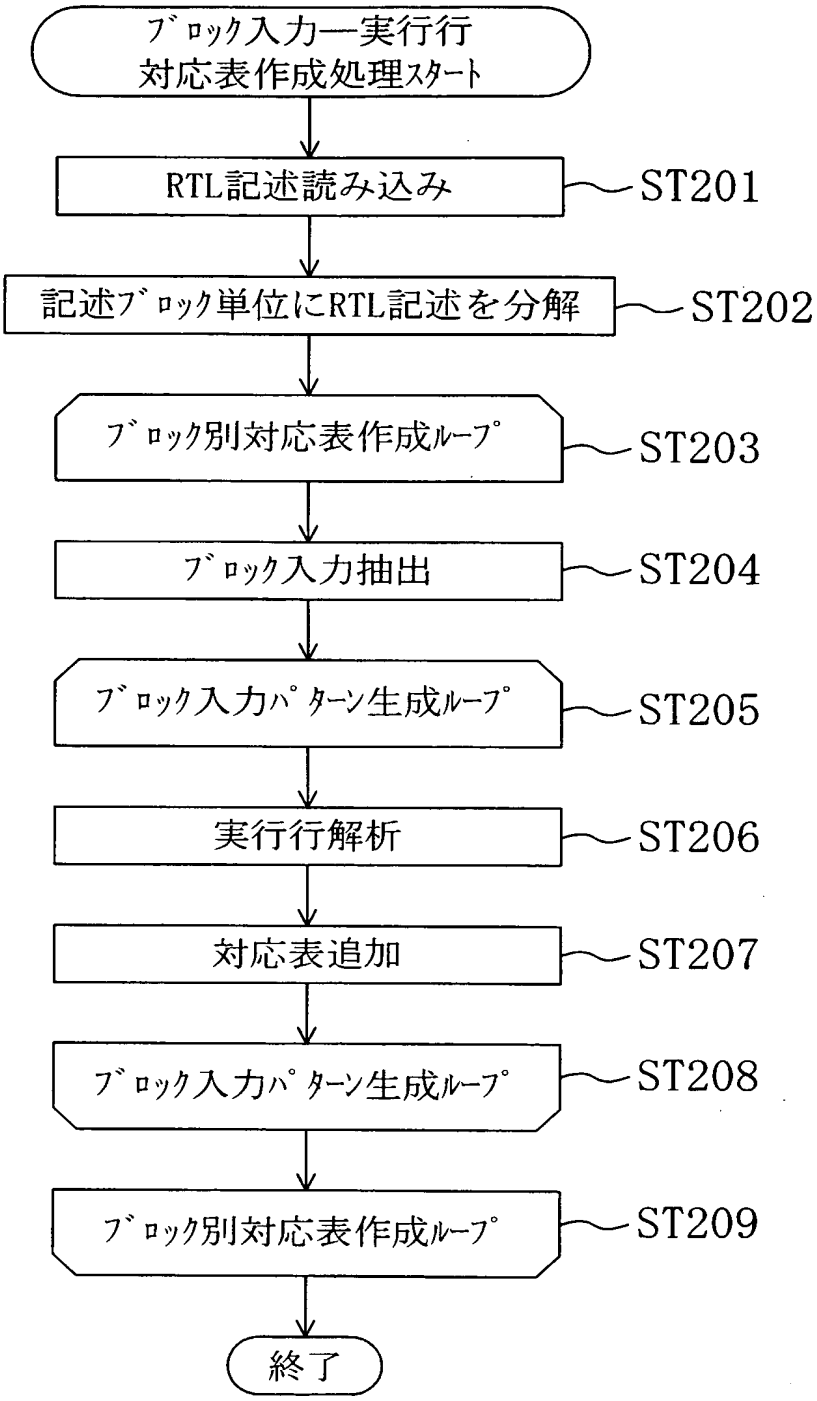
【図４】


6
実行回数
2 100: always @ (a or b or c) begin
2 101: case({a,b,c})
102: 3'b000:out=0;
1 103: 3'b001:out=0;
104: 3'b010:out=0;
105: 3'b011:out=0;
1 106: 3'b100:out=0;
107: 3'b101:out=1;
108: 3'b110:out=1;
109: 3'b111:out=1;
110: endcase
111: end
112:
2 113: always @ (a) begin
2 114: c=~a;
115: end

【図５】


シミュレーション装置
テストベクタ
2
RTL記述
1
ブロック入力—実行行
対応表作成処理部
20
シミュレーション
実行処理部
3
ブロック入力
トレース処理部
22
実行行
解析処理部
23
実行行
集計処理部
12
実行行カウント処理部
21
シミュレーション
結果
4
カバレッジ
結果
6

【図６】

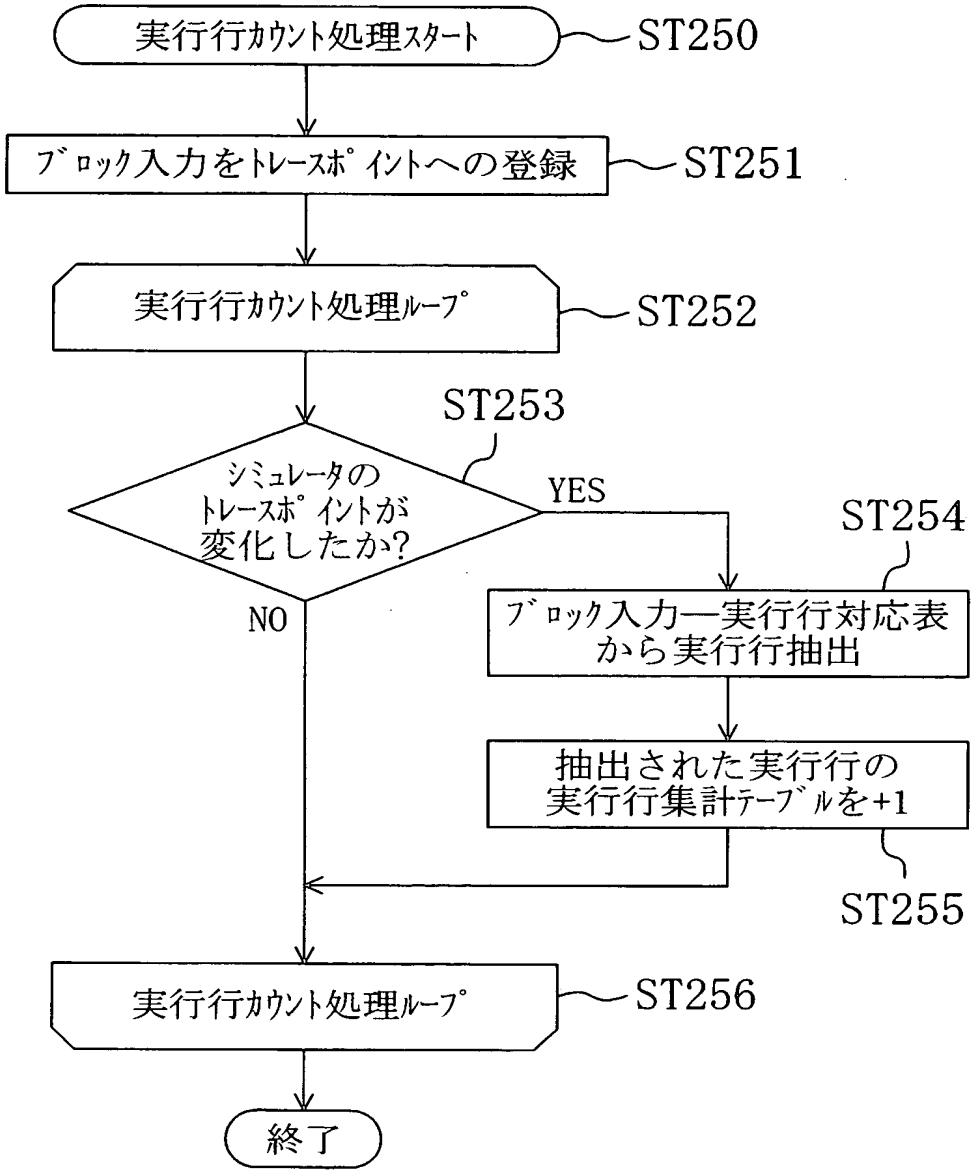

ブロック入力―実行行
対応表作成処理スタート
RTL記述読み込み
ST201
記述ブロック単位にRTL記述を分解
ST202
ブロック別対応表作成ループ
ST203
ブロック入力抽出
ST204
ブロック入力パターン生成ループ
ST205
実行行解析
ST206
対応表追加
ST207
ブロック入力パターン生成ループ
ST208
ブロック別対応表作成ループ
ST209
終了

【図７】

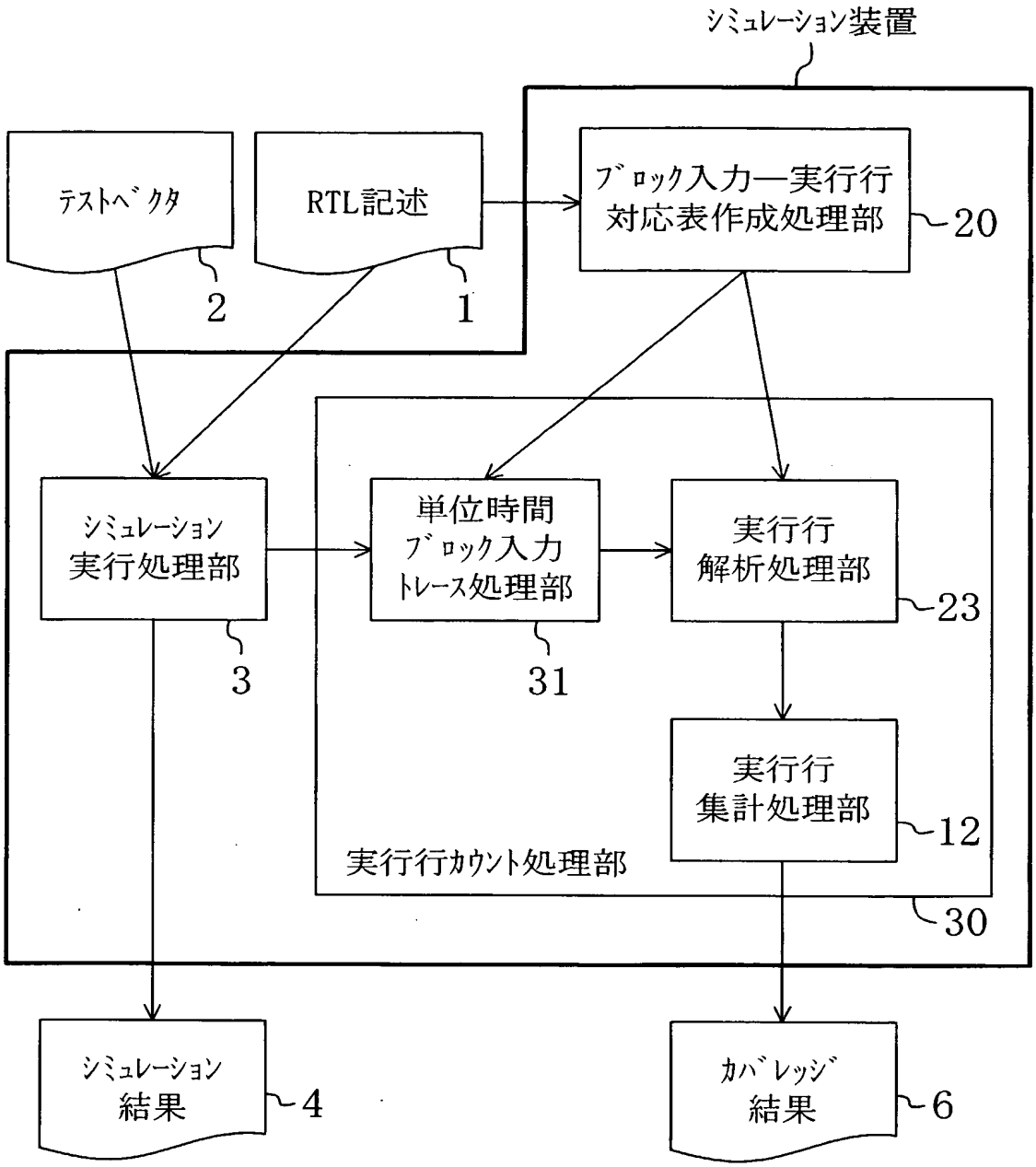
100-110行の記述ﾌﾞﾛｯｸ	
abc	実行行
000	100, 101, 102
001	100, 101, 103
010	100, 101, 104
011	100, 101, 105
100	100, 101, 106
101	100, 101, 107
110	100, 101, 108
111	100, 101, 109

113-115行の記述ﾌﾞﾛｯｸ	
a	実行行
0	113, 114
1	113, 114

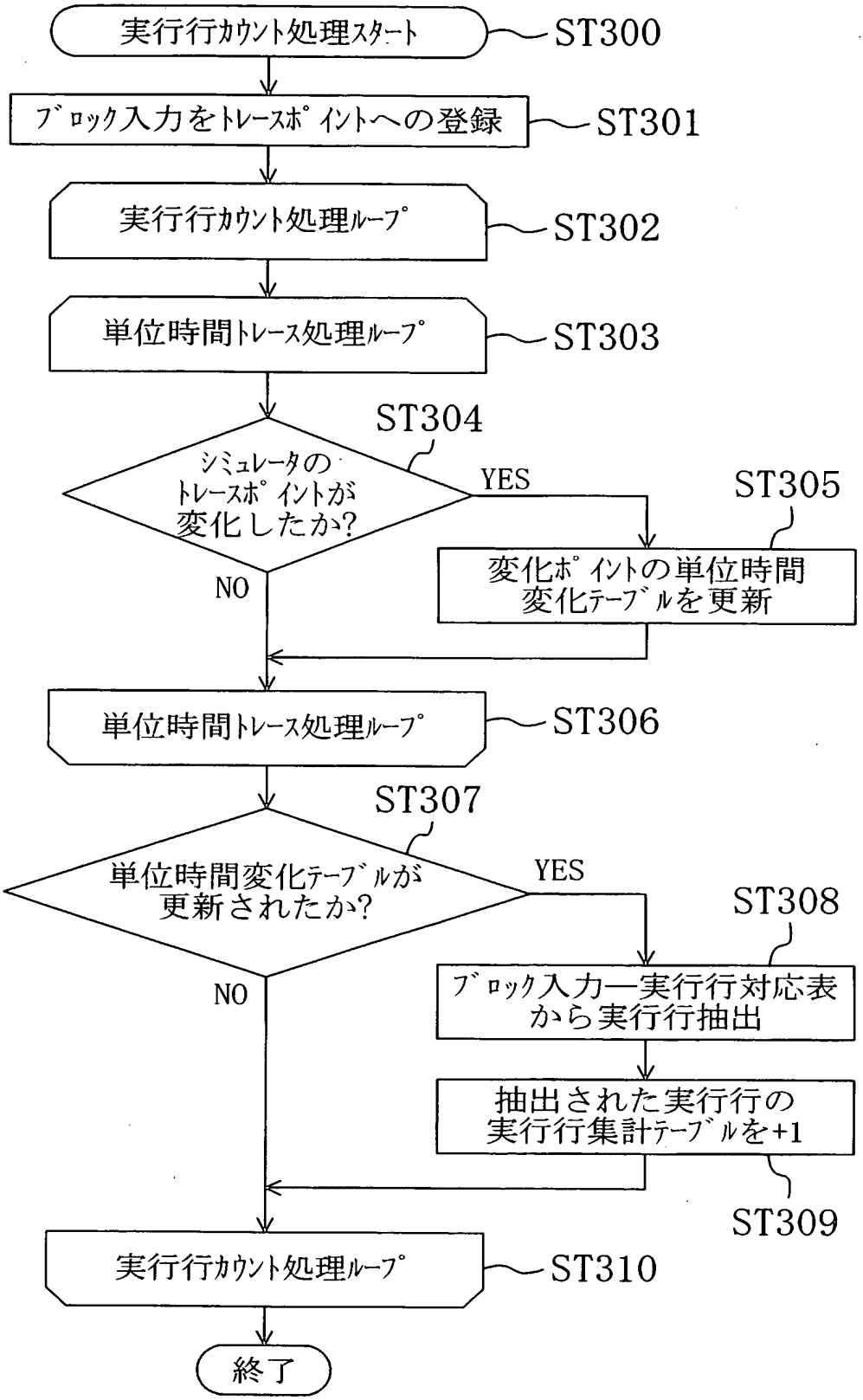
【図８】


実行行カウント処理スタート
ST250
ブロック入力をトレースポイントへの登録
ST251
実行行カウント処理ループ
ST252
ST253
シミュレータの
トレースポイントが
変化したか?
YES
NO
ST254
ブロック入力ー実行行対応表
から実行行抽出
抽出された実行行の
実行行集計テーブルを+1
ST255
実行行カウント処理ループ
ST256
終了

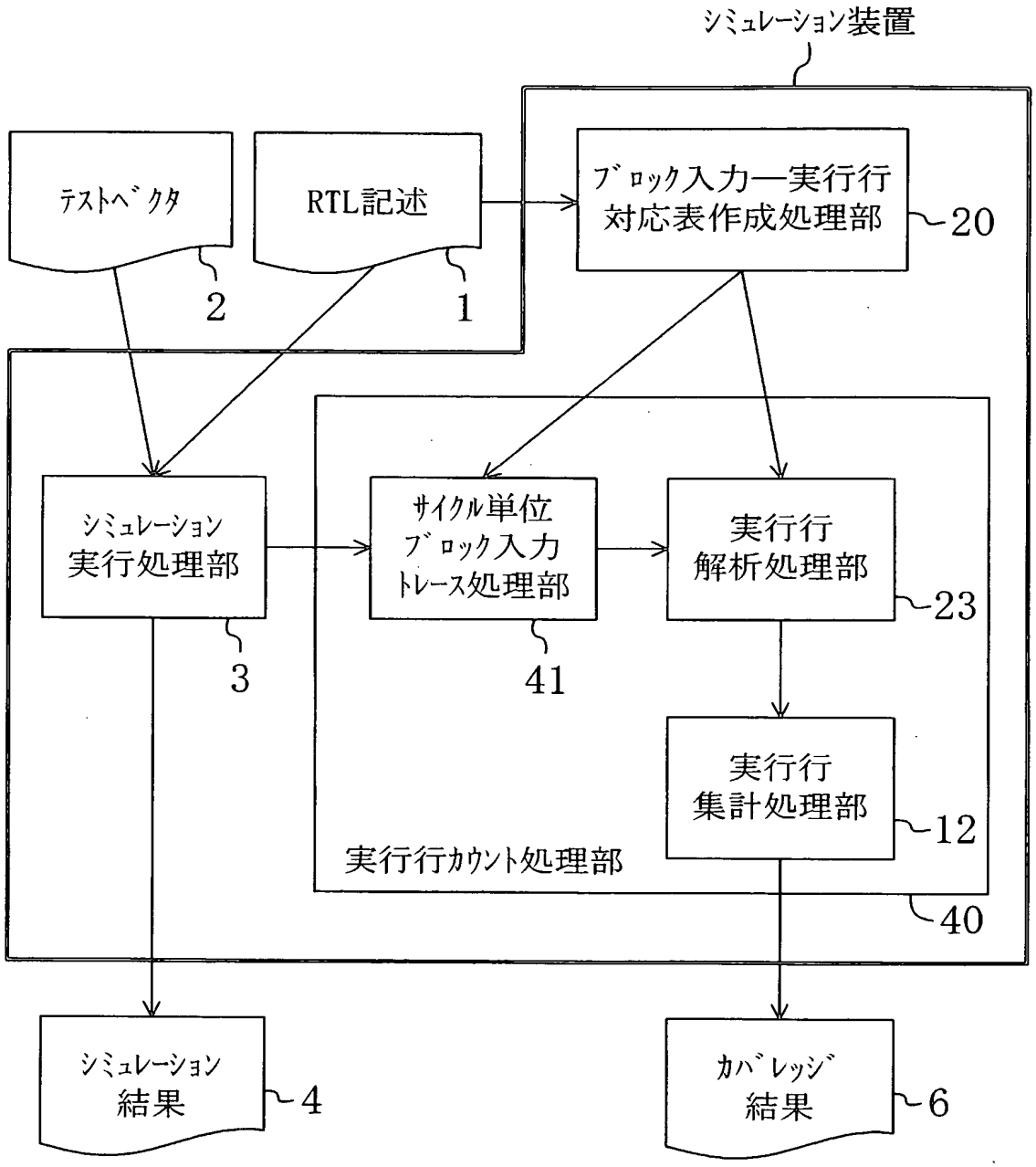
【図９】


シミュレーション装置
テストベクタ
2
RTL記述
1
ブロック入力—実行行
対応表作成処理部
20
シミュレーション
実行処理部
3
単位時間
ブロック入力
トレース処理部
31
実行行
解析処理部
23
実行行
集計処理部
12
実行行カウント処理部
30
シミュレーション
結果
4
カバレッジ
結果
6

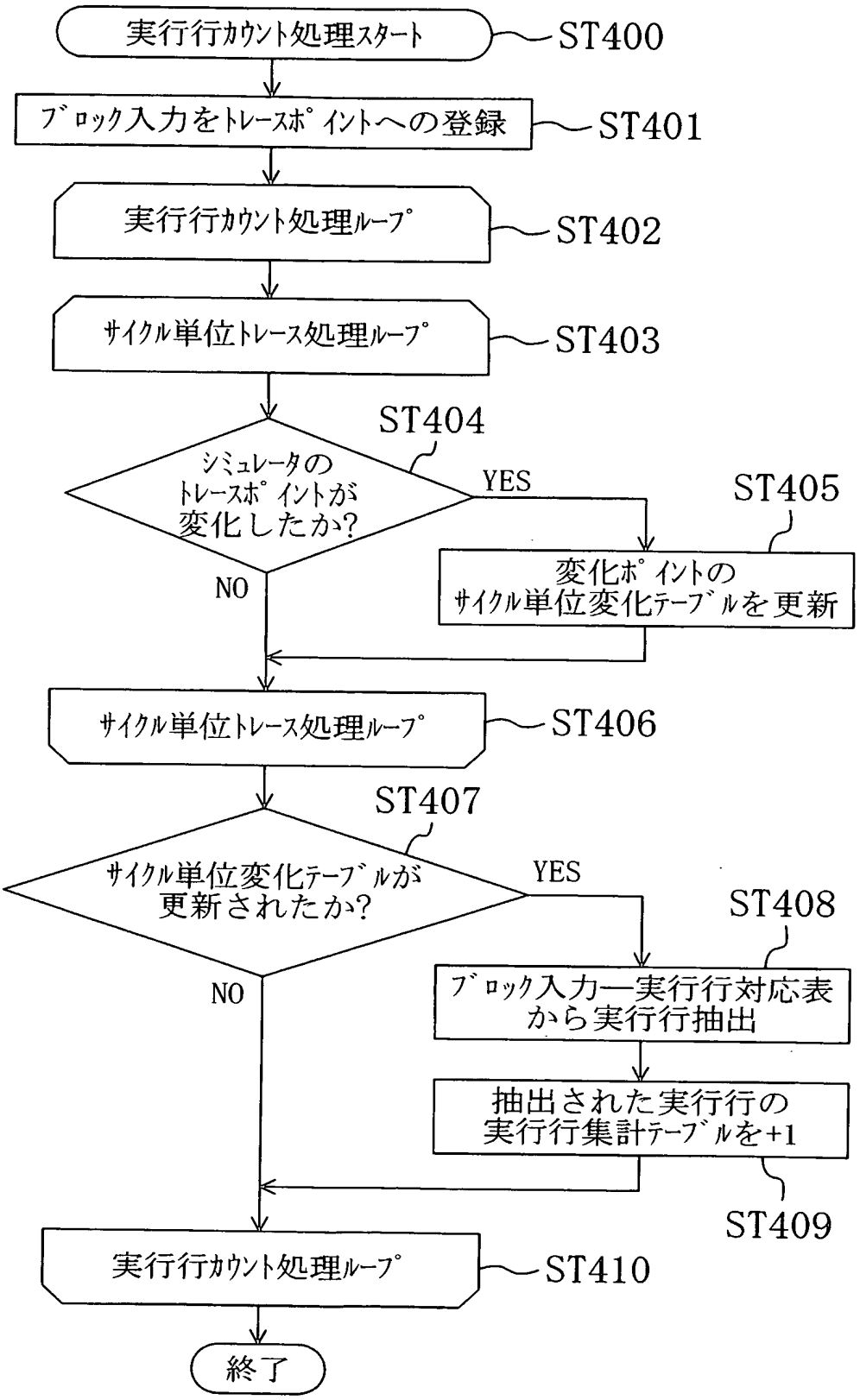
【図１０】


実行行カウント処理スタート
ST300
ブロック入力をトレースポイントへの登録
ST301
実行行カウント処理ループ
ST302
単位時間トレース処理ループ
ST303
ST304
シミュレータの
トレースポイントが
変化したか?
YES
ST305
変化ポイントの単位時間
変化テーブルを更新
NO
単位時間トレース処理ループ
ST306
ST307
単位時間変化テーブルが
更新されたか?
YES
ST308
ブロック入力—実行行対応表
から実行行抽出
NO
抽出された実行行の
実行行集計テーブルを+1
ST309
実行行カウント処理ループ
ST310
終了

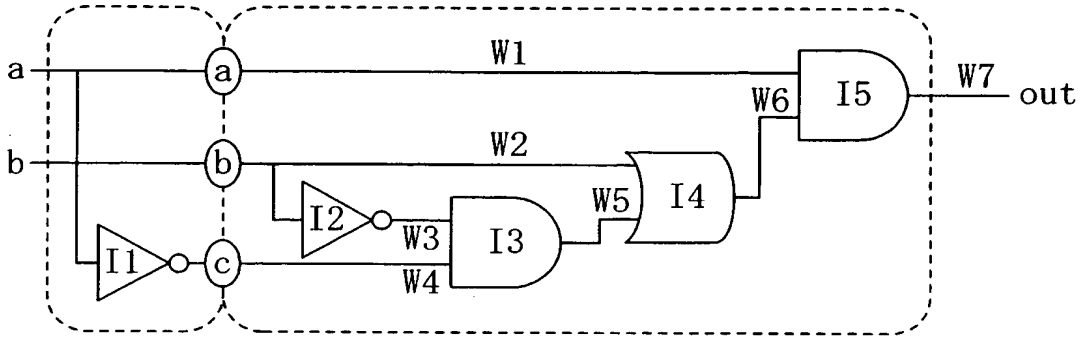
【図１１】


シミュレーション装置
テストベクタ
2
RTL記述
1
ブロック入力ー実行行
対応表作成処理部
20
シミュレーション
実行処理部
3
サイクル単位
ブロック入力
トレース処理部
41
実行行
解析処理部
23
実行行
集計処理部
12
実行行カウント処理部
40
シミュレーション
結果
4
カバレッジ
結果
6

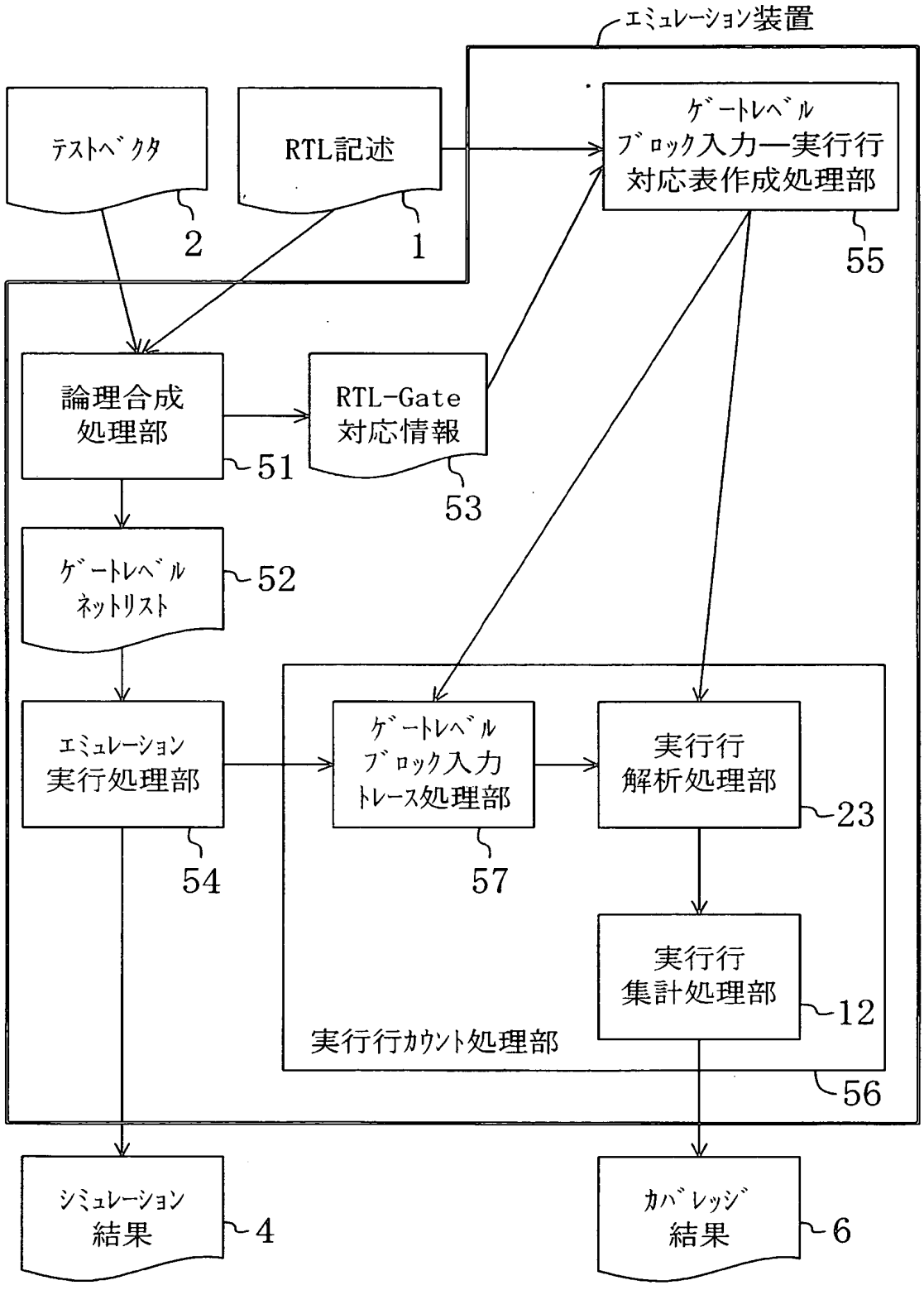
【図１２】


実行行カウント処理スタート
ST400
ブロック入力をトレースポイントへの登録
ST401
実行行カウント処理ループ
ST402
サイクル単位トレース処理ループ
ST403
ST404
シミュレータの
トレースポイントが
変化したか?
YES
ST405
変化ポイントの
サイクル単位変化テーブルを更新
NO
サイクル単位トレース処理ループ
ST406
ST407
サイクル単位変化テーブルが
更新されたか?
YES
ST408
ブロック入力ー実行行対応表
から実行行抽出
NO
抽出された実行行の
実行行集計テーブルを+1
ST409
実行行カウント処理ループ
ST410
終了

【図１３】



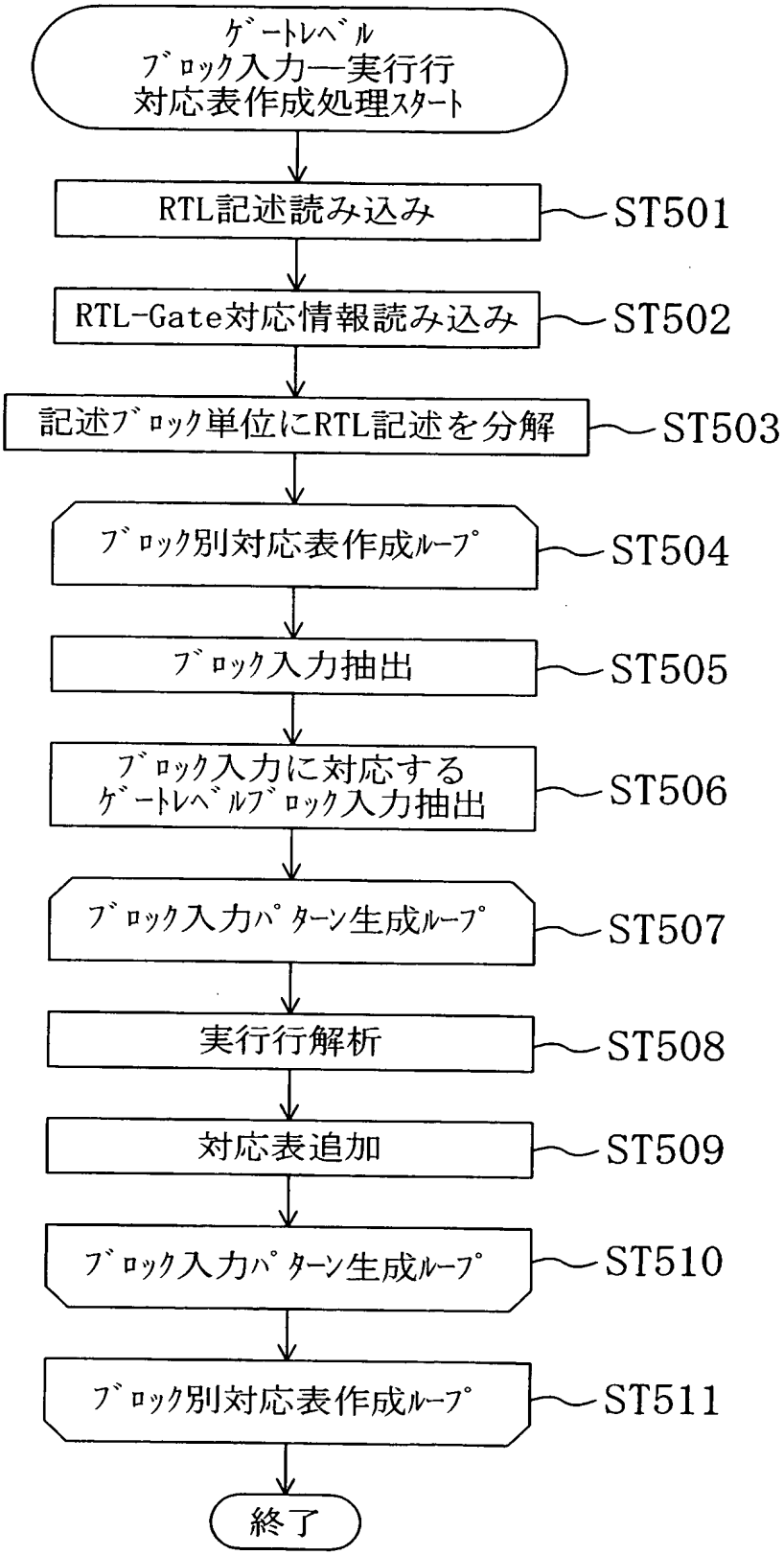
【図１４】


エミュレーション装置
テストベクタ
2
RTL記述
1
ゲートレベル
ブロック入力—実行行
対応表作成処理部
55
論理合成
処理部
51
RTL-Gate
対応情報
53
ゲートレベル
ネットリスト
52
エミュレーション
実行処理部
54
ゲートレベル
ブロック入力
トレース処理部
57
実行行
解析処理部
23
実行行
集計処理部
12
実行行カウント処理部
56
シミュレーション
結果
4
カバレッジ
結果
6

【図１５】

RTL	Gate
a	W1
b	W2
c	W4
out	W7

【図１６】

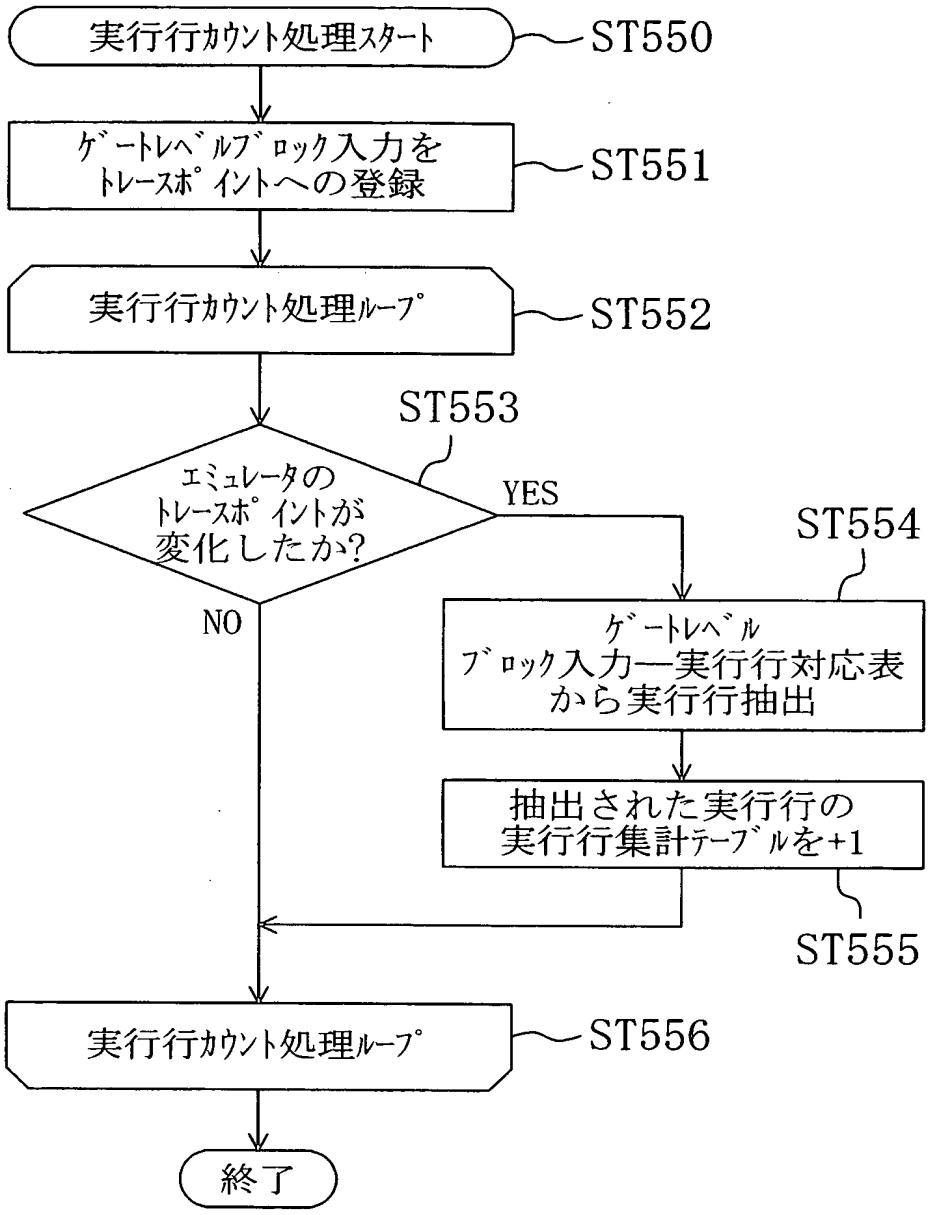

ｹﾞｰﾄﾚﾍﾞﾙ
ﾌﾞﾛｯｸ入力ー実行行
対応表作成処理ｽﾀｰﾄ
RTL記述読み込み
ST501
RTL-Gate対応情報読み込み
ST502
記述ﾌﾞﾛｯｸ単位にRTL記述を分解
ST503
ﾌﾞﾛｯｸ別対応表作成ﾙｰﾌﾟ
ST504
ﾌﾞﾛｯｸ入力抽出
ST505
ﾌﾞﾛｯｸ入力に対応する
ｹﾞｰﾄﾚﾍﾞﾙﾌﾞﾛｯｸ入力抽出
ST506
ﾌﾞﾛｯｸ入力ﾊﾟﾀｰﾝ生成ﾙｰﾌﾟ
ST507
実行行解析
ST508
対応表追加
ST509
ﾌﾞﾛｯｸ入力ﾊﾟﾀｰﾝ生成ﾙｰﾌﾟ
ST510
ﾌﾞﾛｯｸ別対応表作成ﾙｰﾌﾟ
ST511
終了

【図１７】

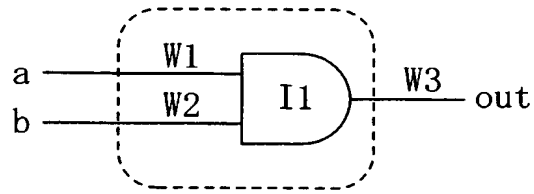
100-110行の記述ﾌﾞﾛｯｸ			
W1	W2	W4	実行行
0	0	0	100, 101, 102
0	0	1	100, 101, 103
0	1	0	100, 101, 104
0	1	1	100, 101, 105
1	0	0	100, 101, 106
1	0	1	100, 101, 107
1	1	0	100, 101, 108
1	1	1	100, 101, 109

113-115行の記述ﾌﾞﾛｯｸ	
W4	実行行
0	113, 114
1	113, 114

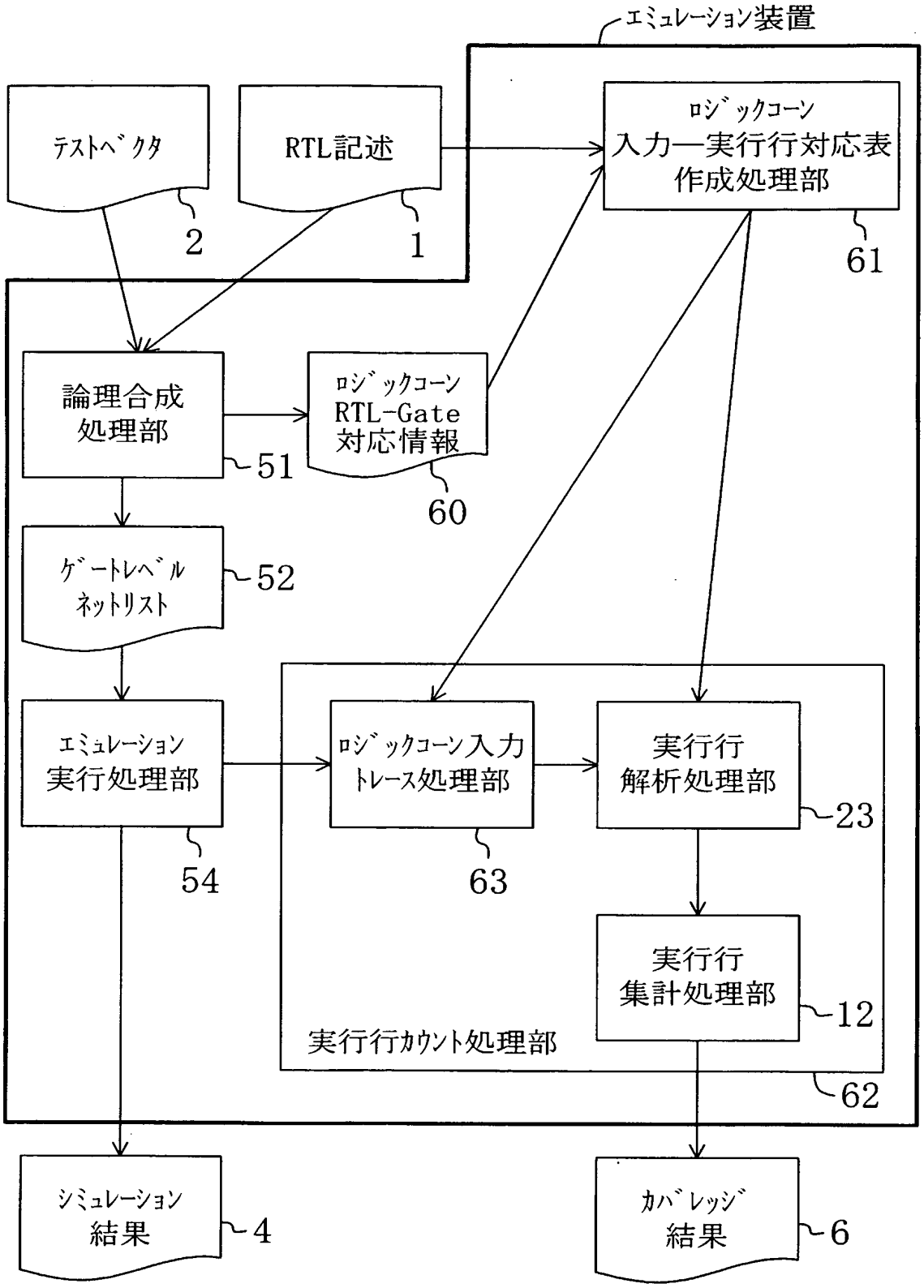
【図１８】


実行行ｶｳﾝﾄ処理ｽﾀｰﾄ
ST550
ｹﾞｰﾄﾚﾍﾞﾙﾌﾞﾛｯｸ入力を
ﾄﾚｰｽﾎﾟｲﾝﾄへの登録
ST551
実行行ｶｳﾝﾄ処理ﾙｰﾌﾟ
ST552
ST553
エミュレータの
トレースポイントが
変化したか?
YES
NO
ST554
ｹﾞｰﾄﾚﾍﾞﾙ
ﾌﾞﾛｯｸ入力ー実行行対応表
から実行行抽出
抽出された実行行の
実行行集計ﾃｰﾌﾞﾙを+1
ST555
実行行ｶｳﾝﾄ処理ﾙｰﾌﾟ
ST556
終了

【図１９】


a
b
W1
W2
I1
W3
out

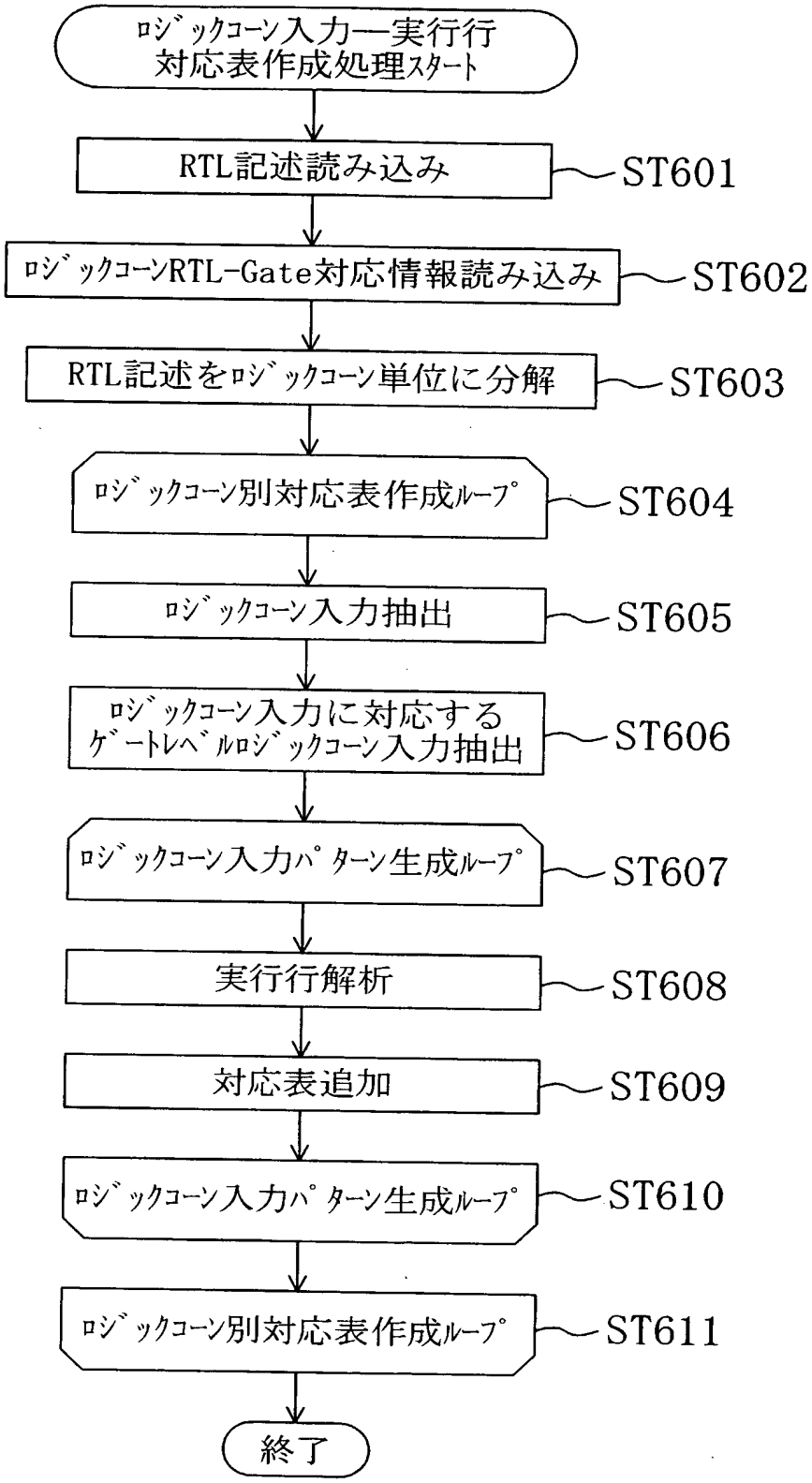
【図２０】


エミュレーション装置
テストベクタ
2
RTL記述
1
ロジックコーン
入力ー実行行対応表
作成処理部
61
論理合成
処理部
51
ロジックコーン
RTL-Gate
対応情報
60
ゲートレベル
ネットリスト
52
エミュレーション
実行処理部
54
ロジックコーン入力
トレース処理部
63
実行行
解析処理部
23
実行行
集計処理部
12
実行行カウント処理部
62
シミュレーション
結果
4
カバレッジ
結果
6

【図２１】

RTL	Gate
a	W1
b	W2

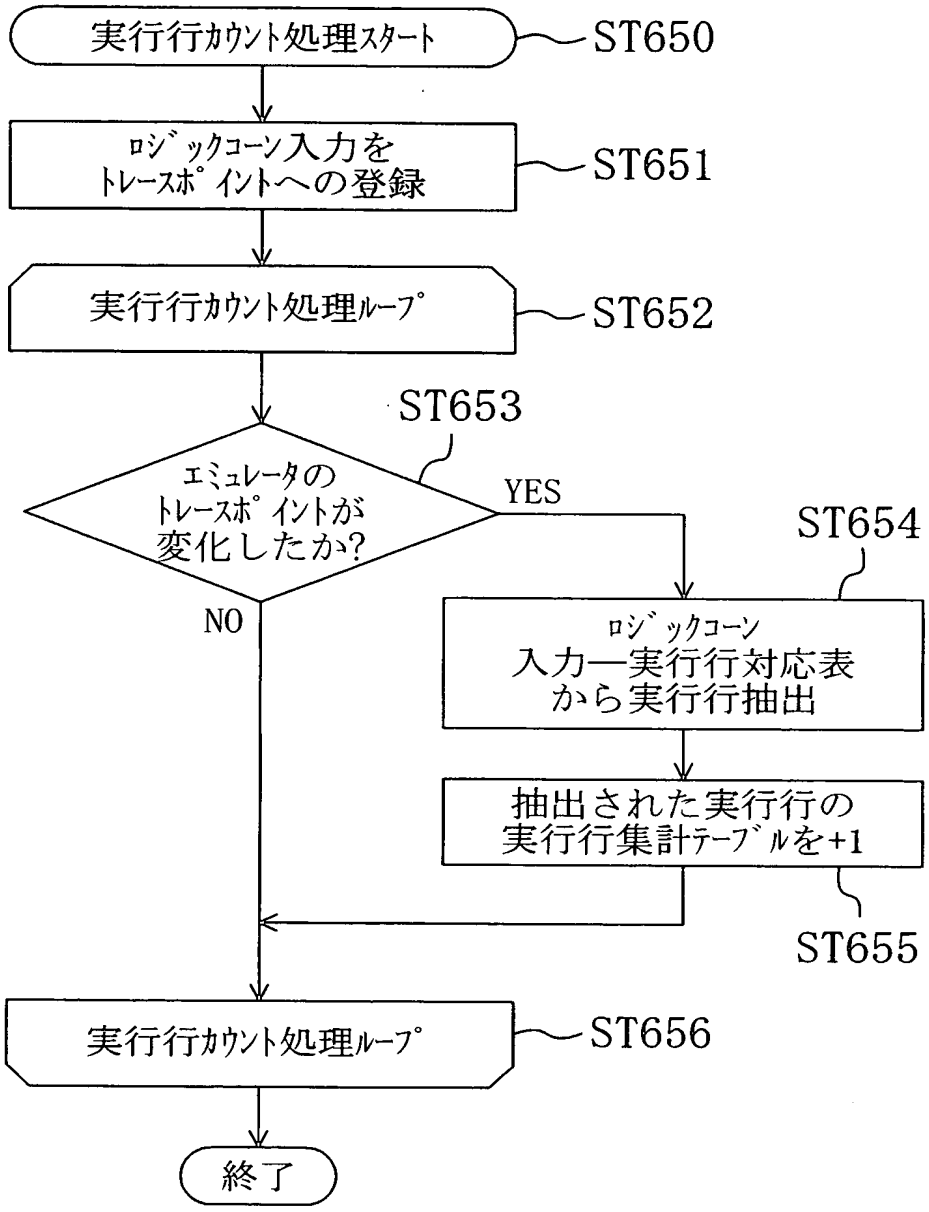
【図２２】


ﾛｼﾞｯｸｺｰﾝ入力ー実行行
対応表作成処理ｽﾀｰﾄ
RTL記述読み込み
ST601
ﾛｼﾞｯｸｺｰﾝRTL-Gate対応情報読み込み
ST602
RTL記述をﾛｼﾞｯｸｺｰﾝ単位に分解
ST603
ﾛｼﾞｯｸｺｰﾝ別対応表作成ﾙｰﾌﾟ
ST604
ﾛｼﾞｯｸｺｰﾝ入力抽出
ST605
ﾛｼﾞｯｸｺｰﾝ入力に対応する
ｹﾞｰﾄﾚﾍﾞﾙﾛｼﾞｯｸｺｰﾝ入力抽出
ST606
ﾛｼﾞｯｸｺｰﾝ入力ﾊﾟﾀｰﾝ生成ﾙｰﾌﾟ
ST607
実行行解析
ST608
対応表追加
ST609
ﾛｼﾞｯｸｺｰﾝ入力ﾊﾟﾀｰﾝ生成ﾙｰﾌﾟ
ST610
ﾛｼﾞｯｸｺｰﾝ別対応表作成ﾙｰﾌﾟ
ST611
終了

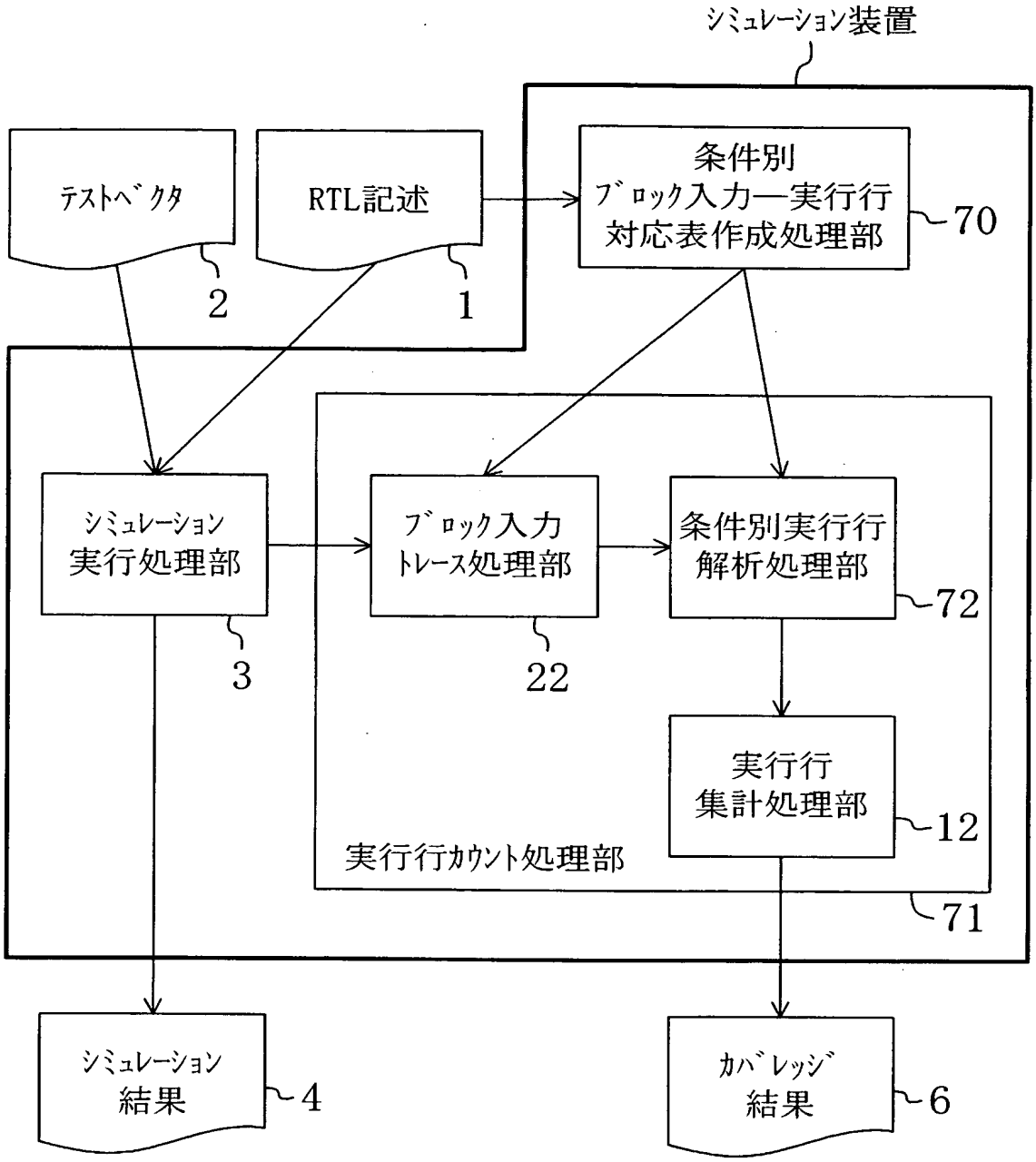
【図２３】

W1	W2	実行行
0	0	100, 101, 103, 113, 114
0	1	100, 101, 104, 113, 114
1	0	100, 101, 106, 113, 114
1	1	100, 101, 108, 113, 114

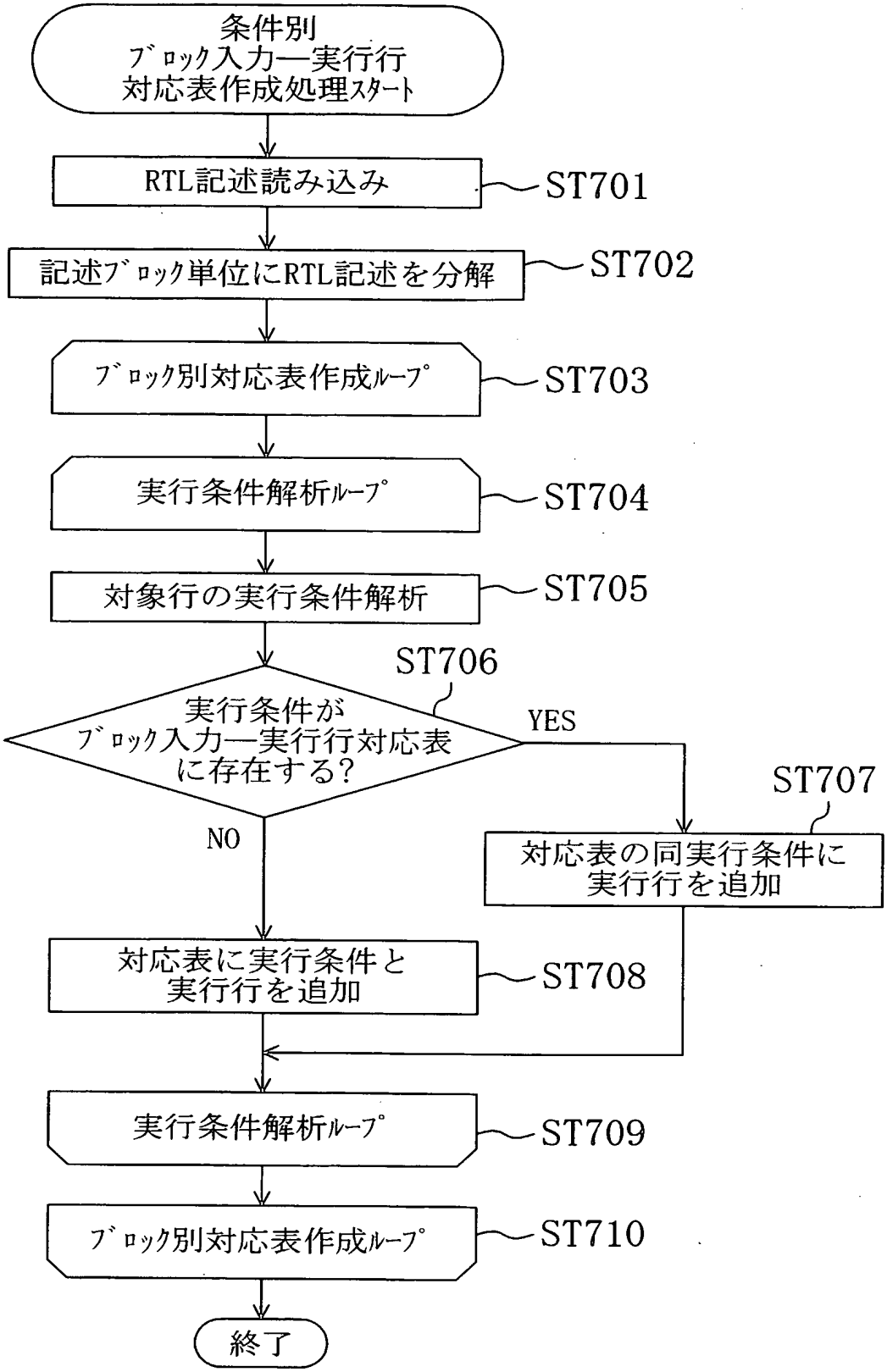
【図２４】



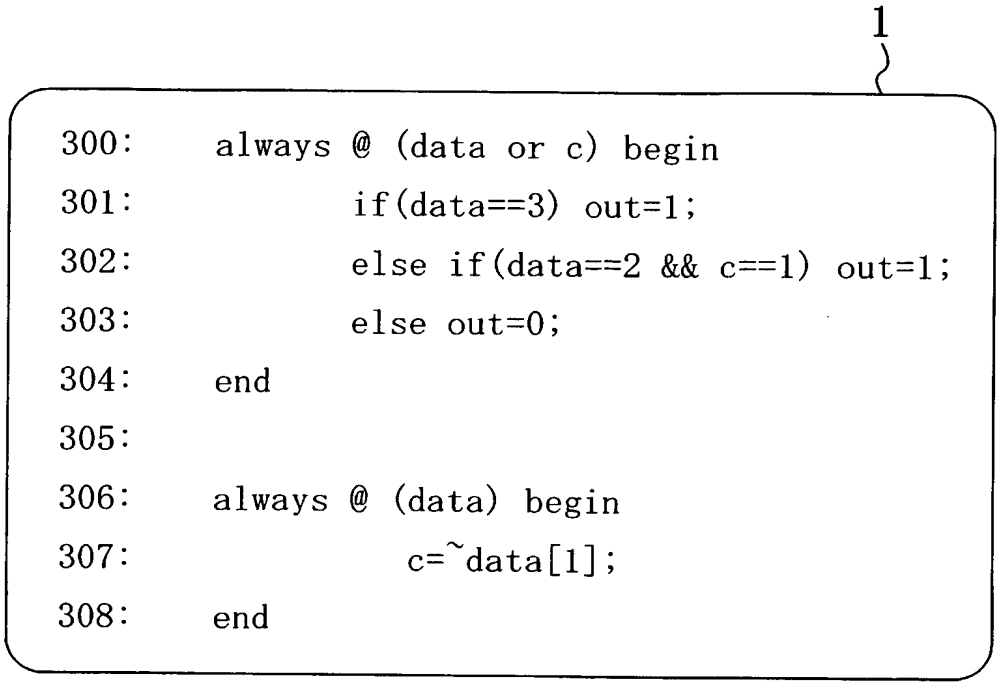
【図２５】


シミュレーション装置
テストベクタ
2
RTL記述
1
条件別
ブロック入力ー実行行
対応表作成処理部
70
シミュレーション
実行処理部
3
ブロック入力
トレース処理部
22
条件別実行行
解析処理部
72
実行行
集計処理部
12
実行行カウント処理部
71
シミュレーション
結果
4
カバレッジ
結果
6

【図２６】


条件別
ブロック入力―実行行
対応表作成処理スタート
RTL記述読み込み
ST701
記述ブロック単位にRTL記述を分解
ST702
ブロック別対応表作成ループ
ST703
実行条件解析ループ
ST704
対象行の実行条件解析
ST705
ST706
実行条件が
ブロック入力―実行行対応表
に存在する?
YES
NO
ST707
対応表の同実行条件に
実行行を追加
対応表に実行条件と
実行行を追加
ST708
実行条件解析ループ
ST709
ブロック別対応表作成ループ
ST710
終了

【図２７】

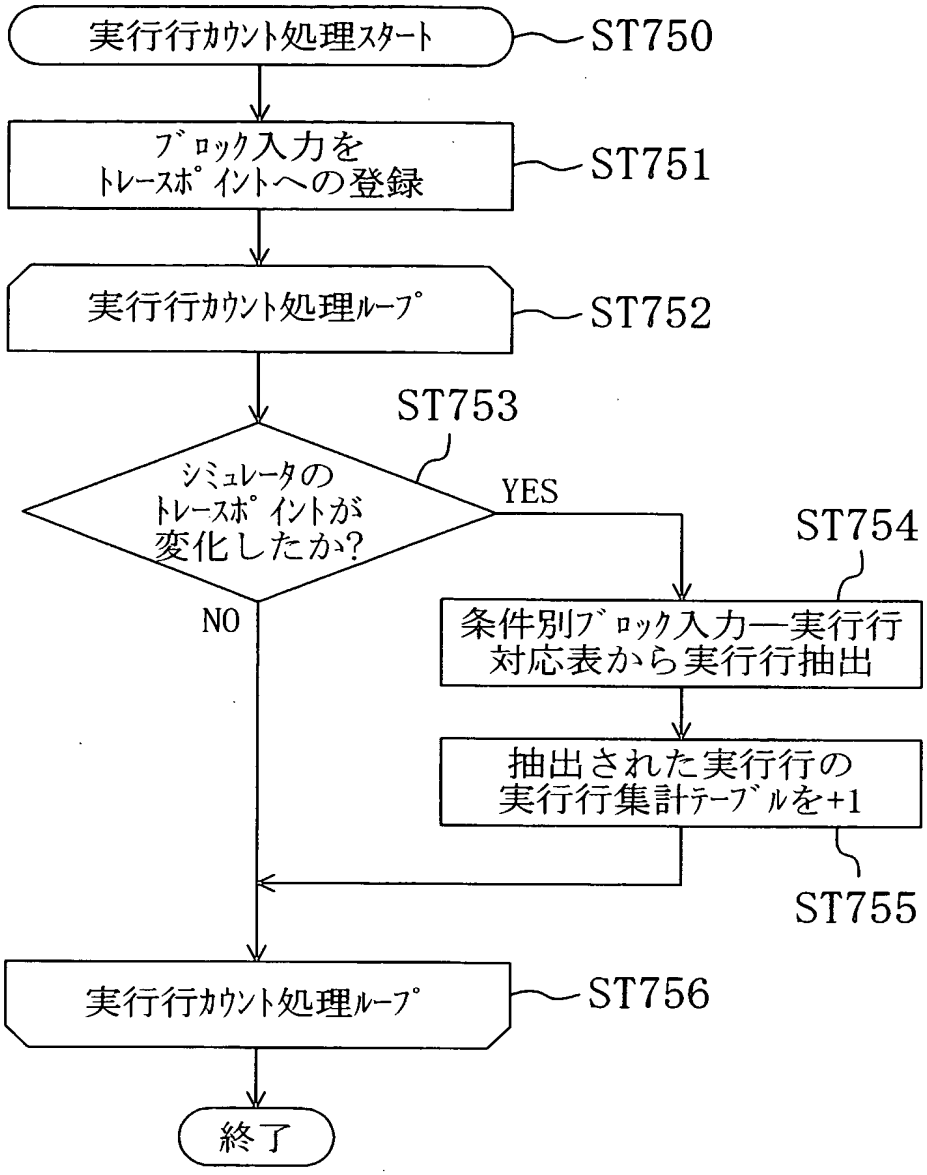


【図２８】

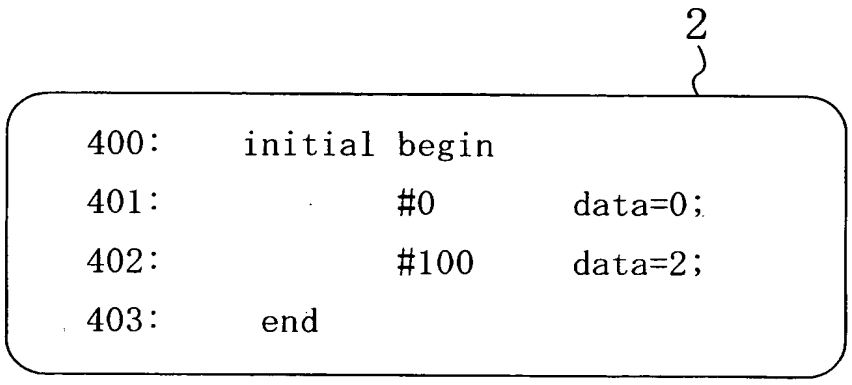
300-304行の記述ﾌﾞﾛｯｸ	
入力条件	実行行
data=3	300, 301
!(data=3)&(data=2)&(c=1)	300, 302
!(data=3)&!(data=2)&!(c=1)	300, 303

306-308行の記述ﾌﾞﾛｯｸ	
入力条件	実行行
ALL	306, 307

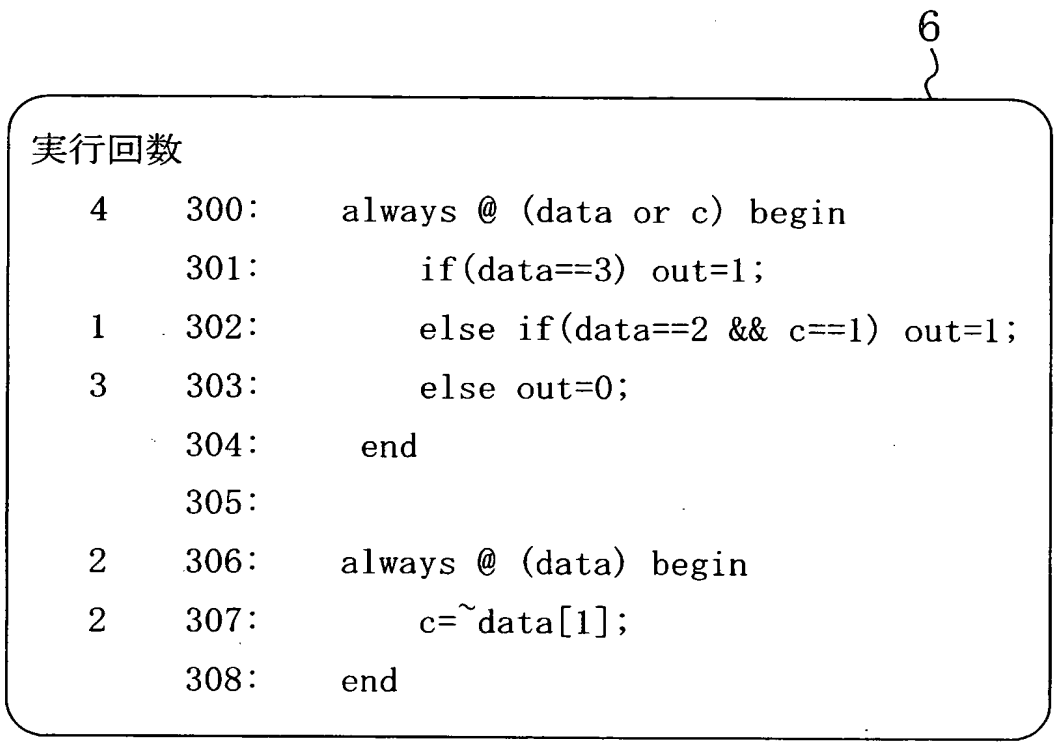
【図２９】


実行行カウント処理スタート
ST750
ブロック入力を
トレースポイントへの登録
ST751
実行行カウント処理ループ
ST752
ST753
シミュレータの
トレースポイントが
変化したか?
YES
NO
ST754
条件別ブロック入力—実行行
対応表から実行行抽出
抽出された実行行の
実行行集計テーブルを+1
ST755
実行行カウント処理ループ
ST756
終了

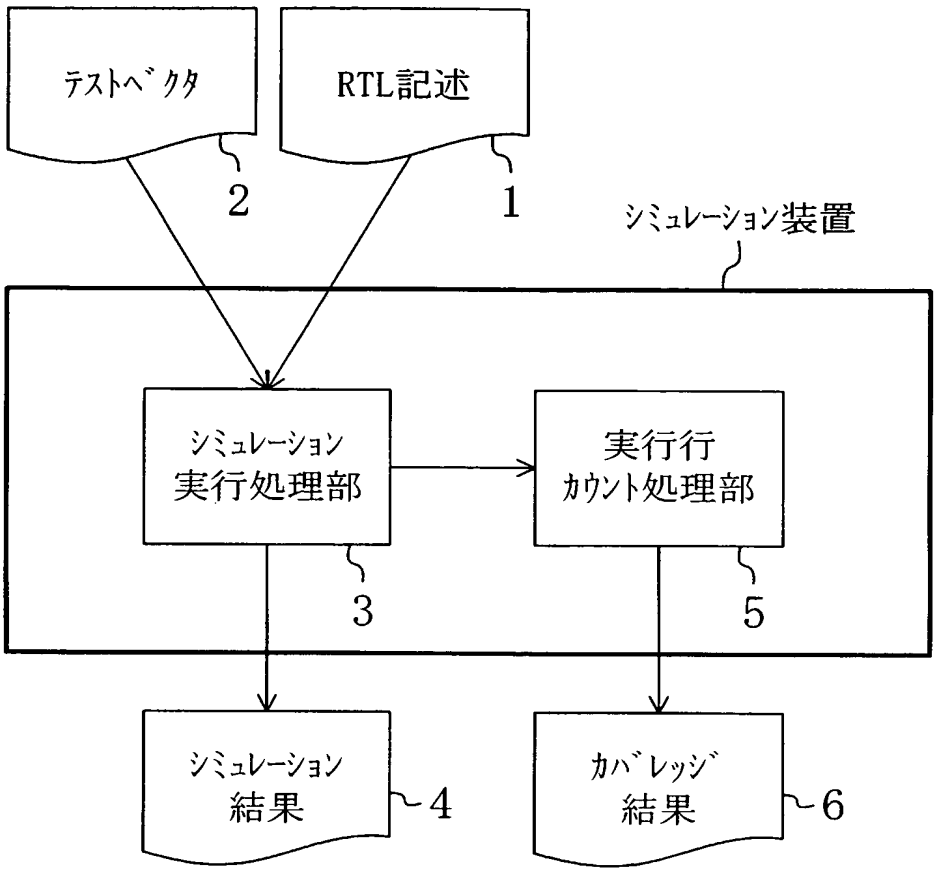
【図３０】


2
400:　initial begin
401:　　#0　　data=0;
402:　　#100　　data=2;
403:　end

【図３１】


6
実行回数
4　300:　always @ (data or c) begin
301:　　if(data==3) out=1;
1　302:　　else if(data==2 && c==1) out=1;
3　303:　　else out=0;
304:　end
305:
2　306:　always @ (data) begin
2　307:　　c=~data[1];
308:　end

【図３２】


ﾃｽﾄﾍﾞｸﾀ
RTL記述
2
1
ｼﾐｭﾚｰｼｮﾝ装置
ｼﾐｭﾚｰｼｮﾝ
実行処理部
実行行
ｶｳﾝﾄ処理部
3
5
ｼﾐｭﾚｰｼｮﾝ
結果
4
ｶﾊﾞﾚｯｼﾞ
結果
6

【図３３】

実行回数

```
4     100:    always @ (a or b or c) begin
4     101:            case({a,b,c})
      102:                    3'b000:out=0;
1     103:                    3'b001:out=0;
      104:                    3'b010:out=0;
      105:                    3'b011:out=0;
1     106:                    3'b100:out=0;
1     107:                    3'b101:out=1;
      108:                    3'b110:out=1;
      109:                    3'b111:out=1;
      110:            endcase
      111:    end
      112:
2     113:    always @ (a) begin
2     114:             c=~a;
      115:    end
```

【書類名】　要約書

【要約】

【課題】　ハードウェア記述言語検証時のソースコードカバレッジ測定において、シミュレーション実行処理が逐次処理である為、機能的に実行されないはずの行が瞬間的に実行され、不当に網羅性の高いカバレッジ結果が出力されるという問題があった。

【解決手段】ハードウェア記述の記述ブロック単位に実行行履歴を記録し、同一時刻に同じ記述ブロックが実行された場合、同一時刻の前回の実行履歴を消去する工程を有し、カバレッジ測定時の誤カウントを防止する。

【選択図】図１

特願２００２－３５５８１９

出　願　人　履　歴　情　報

識別番号　　［０００００５８２１］

１．変更年月日　　１９９０年　８月２８日
［変更理由］　　新規登録
住　所　　大阪府門真市大字門真１００６番地
氏　名　　松下電器産業株式会社